ЎЗБЕКИСТОН АЛОҚА ВА АХБОРОТЛАШТИРИШ АГЕНТЛИГИ

ТОШКЕНТ АХБОРОТ ТЕХНОЛОГИЯЛАРИ УНИВЕРСИТЕТИ

УРГАНЧ ФИЛИАЛИ

Рахимов Б.С.

ПАРАЛЛЕЛ ХИСОБЛАШ УСУЛЛАРИ

ФАНИДАН

МАЪРУЗА МАТНЛАРИ

Урганч 2007

Мундарижа: Бет

Кириш

Ихтиёрий бир одам қаердадир афсонавий “жуда қувватли” компьютер борлиги хақида эшитиши эхтимолдан холи эмас. Хақиқатдан хам яқиндагина Россиянинг Гидрометео марказига Cray Research фирмасининг қувватли компьютерлари ўрнатилди. 1999 йил ноябрида корхоналараро суперкомпьютер маркази расмий равишда очилди. Бу марказда хозирги пайтда 786 та процессорли компьютер мавжуд. Компьютер саноатининг тез ривожланиши ушбу тушунча нисбий эканлигини вужудга келтирди. Чунки 10 йил олдин уни суперкомпьютер дейиш мумкин эди, лекин хозирги кунда бу компьютерларни суперкомпьютер деб аташ унчалик тўғри эмас. Ихтиёрий компьютернинг асосий параметрлари бир-бири билан чамбарчас боғлиқ. Юқори тезликка ва кичик аператив хотирага, ёки катта апреративхотирага ва кичик хажмли дискка эга бўлган универсал компьютерни тасаввур этиш қийин. Бундан қуйидаги хулосани чиқарамиз: супер-ЭХМ бу хозирги кундаги фақат юқори унумдорликка эмас, балки максимал аператив хотира ва диск хажмига эга бўлган компьютер. Супер-ЭХМ ва ўта юқори унумдорлик нима учун зарур? Суперкомпьютерлар фақат нефтни қайта ишлаш саноатида эмас, балки бошқа сохаларда хам қўлланилади. Инсон фаолият юритадиган суперкомпьютерлар зарур бўлган сохаларга автомобилсозлик, нефт ва газ қазиш, фармакология, об-ҳаво ўзгаришини моделлаш маркази, зилзилавий изланиш, электрон қурилмаларни лойихалаш, янги материалларни синтезлаш ва бошқалар мисол бўлади. Нима учун суперкопьютерлар тез хисоблайди? Бу саволга турлича вариантларда жавоб топиш мумкин. Бу жавоблардан иккитаси хақиқатга яқин, яъни элементлар базасини ривожланиши ва компьютер архитектурасига киритилган янгиликлар жавоб бўлади. Келтирилган жавоблардан қайси бири рекорд унумдорликка эришиш учун тўғри ечим бўлиши мумкинлигини тахлил қилиб кўрамиз. 1946 йил Кембриджда яратилган ер юзида биринчи

компьютерлардан бири EDSAC компьютери 2 микросекунд вақт тактига эга. Унда 18*n миллисекундда 2*n та арифметик амал бажарилган, яъни бир секунда ўртача 100 та арифметик амал бажарилган.

Уни хозирги кундаги суперкомпьютерлардан бири Hewlett-Paskard V2600 билан солиштирсак. Бу компьютерларда вақт такти 1.8 наносекунд, энг юқори унумдорлиги секундига 77миллиард арифметик амалга тенг. Солиштириш натижаси қандай бўлади? Ярим аср давомида компьютерлар унумдорлиги 700 мартадан кўпроққа ошган. Бунда тезликдаги ютуқ такт вақтини 2 микросекунддан 1.8 наносекундга тушган. Бу 1000 марта камайган. Бу ва бошқа ютуқларга қандай эришилган? Жавоб маълум компьютер архитектурасида қўлланилган янгиликлар ёрдамида эришилган. Бу янгиликлар ичида маълумотларни параллел хисоблаш принципи асосий ўринда туради. Параллел хисоблаш асосида бир неча амалларни бир вақтда бажариш (параллел хисоблаш) ғояси ётади.

1.1. Маълумотларни параллел қайта ишлаш

Бир неча амалларни бир вақтда бажариш ғоясидан иборат бўлган маълумотларни параллел хисоблаш икки хил кўриниши мавжуд.

Булар: Параллел ва конвейер.

Агар бирор қурилма битта амални вақт бирлигида бажарса, у ҳолда мингта амални минг вақт бирлигида бажаради. Агар худди шундай бир вақтда ишлай оладиган ва бир–бирига мустақил бешта қурилма мавжуд деб қаралса, у ҳолда улар юқоридаги мингта амални мингта вақт бирлигида эмас, балки икки юзта вақт бирлигида бажаради. Худди шундай N та қурилмадан иборат тизим 1000 та амални 1000/N вақт бирлигида бажарида. Унга ўхшаш ҳолатларни ҳаётдан ҳам келтириш мумкин. Масалан, агар битта аскар полизга 10 соатда ишлов берса, у холда 50 аскардан иборат рота бир вақтда ишлаб полизга 12 минутда ишлов беради. Бу параллел амаллар принципи

ҳисобланади.

Конвейерли қайта ишлаш.

Қўзғалувчан вергулли шаклда тасвирланган хақиқий иккита сонни қўшиш учун нима қилиш керак? Бунда бир қатор майда амаллар бажарилади. Булар: тартибини солиштириш, тартибини тенглаш, нормаллаш ва бошқа амаллар. Дастлабки компьютерларнинг процессорлари юқорида келтирилган барча “микро амалларни” ҳар бир аргументлар жуфтлиги учун натижани хосил қилгунча кетма-кет бажарган ва бундан кейин қўшилувчиларнинг кейинги жуфтлигини қайта ишлашга ўтган. Конвейерли қайта ишлаш ғоясида умумий амал бир неча босқичларга ажратилади. Ҳар бир босқич бажарилгандан кейин унинг натижаси кейинги босқичга берилади ва шу билан бирга кирувчи маълумотларнинг янги қисми қабул қилинади. Бунда олдин бажарилган амалларни натижаларини қўллаш орқали қайта ишлаш тезлиги оширилади. Фараз қилайлик, амал бешта микро амалдан иборат бўлиши ва уларни ҳар бири битта вақт бирлигида бажаради. Агар ажралмас ягона кетма-кет курилма мавжуд бўлса, у 100 та аргументлар жуфтлигини 500 вақт бирлигида бажаради. Агар ҳар бир микро амал конвейрли қурилманинг алоҳида босқичида бажарилса, у холда бундай қурилманинг ҳар бир қайта ишлаш босқичининг бешинчи вақт бирлигида биринчи 5та аргументлари аниқланади. Биринчи натижа вақтнинг 5-бирлигидан кейин олинади. 100 та жуфтликдан иборат тўплам эса 5+99=104 вақт бирлигидан кейин олинади. Яъни параллел қурилмага нисбатан 5 марта тез бажарилади. Бир қарашда конвейерли қайта ишлашни параллел қурилмаларини ўрнига зарур миқдордаги конвейр қурилмаларини қўллаш мумкиндек кўринади. Бироқ бунда хосил бўлган система нархи ва мураккаблиги ошади. Унумдорлик эса ўзгармай қолади. Параллел дастур тузиш учун, дастурдаги бир вақтда ва бир-биридан мустақил процессорларда бажариладиган амаллар гуруҳини ажратиб олиш керак. Бунинг имконияти мавжудлиги дастурда

информатцион боғлиқликлар мавжудлиги ёки йўқлиги билан аниқланади. Агар дастурнинг бирор амали натижаси иккинчи амал аргументи сифатида қўлланилса амаллар информацион боғлиқ деб аталади. Агар В амали А амалига информацион боғлиқ бўлса, у ҳолда В амали фақт А амали тугагандан кейин бажарилади. Агар А ва В амаллари информацион боғлиқмас бўлса, у холда алгоритмда уларни бажариш кетам-кетлигига чекланиш қўйилмайди, хусусан улар бир вақтда бажарилиши мумкун. Шундай қилиб, дастурни информацион боғлиқ амалларни аниқлашдан ва уларни хисоблаш қурилмаларига тақсимлашдан, синхронлашдан ва зарур коммуникацияни ўрнатишдан иборат бўлади.

1.2. Маълумотларни алмашиш ва қайта ишлаш усуллари.

Массивли-параллел компьютерлар пайдо бўлиши билан параллел жараёнларни алоқасини қўлаб-қувватловчи кутубхона интерфейслар кенг тарқалди. Бу йўналишнинг тоифали вакилига Message Passing Interface (MPI) интерфейси мисол бўлади. Бу интерфейс амалда вектор-конвейерли супер - ЭХМ дан тортиб шахсий компьютергача бўлган барча параллел платформаларда мавжуд. Дастурнинг қайси параллел жараёнлари дастурнинг қайси қисмида ва жараёнлар билан маълумотлар алмашиши ёки ўз ишини синхронлаб бориши кераклигини дастурчининг ўзи белгилайди. Одатда параллел жараёнларнинг манзил сохаси турлича бўлади. Хусусан, бу ғояга MPI ва PVI да амал қилинади. Бошқа технологияларда, масалан Shmem да локал (private) ва умумий (shared) ўзгарувчилари қўлланилади. Бу ўзгарувчиларга дастурнинг барча жараёнлари мурожат этиши мумкин ва Put/Get тоифасидаги операциялар ёрдамида умумий хотира билан ишлаш усули ташкил этилади. Linda системаси ўзига хос хусусиятга эга бўлиб, унда ихтиёрий кетма-кетликда тилга тўртта: in, out, read ва eval функцияларини қўшади ва параллел дастурлар тузиш имконини беради. Афсуски, ушбу

келтирилган ғоя оддий бўлишига қарамасдан, уни амалда қўллаш муаммолар туғдиради.

1.3.Хабарларни алмашиш интерфейс технологияси

Тақсимланган хотира параллел компьютерлардаги кенг тарқалган дастурлаш технологияси MPI технологияси хисобланади. Бундай системалардаги параллел жараёнларнинг ўзаро мулоқат усули бир-бири билан хабарлар алмашишдан иборат. Бу усул технологияси номи - Message Passing Interface (хабарлар алмашиш интерфейси) деб акс эттирилган. MPI стандартида система амал қилиши ва дастур яратишда фойдаланувчилар амал қилиши керак бўлган қоидалар мавжуд. MPI Фортран ва Си билан ишлашни қўллаб-қувватлайди. Интерфейснинг тўлиқ версиясида 125 тадан кўпроқ процедура ва функциялар мавжуд. MPI MIMD(Multiple Instruction Multiple Data) стилидаги параллел дастурларни қўллаб қувватлайди. Унда турли матнлар билан берилган жараёнлар бирлаштирилади. Бироқ бундай дастурларни тузиш ва отладка этиш мураккаб. Шунинг учун амалда дастурчилар SPMD (SINGLE PROGRAM MULTIPLE DATA) параллел дастурлаш моделидан фойдаланишади. Унда параллел жараёнлар учун битта дастур коди қўлланилади. Хозирги пайтда MPI кўпроқ тўрлар билан ишлашни қўллаб-қувватлайди. MPI кутубхонасининг параллел жараёнлар билан мўлжалланган функцияларини қўллаб-қувватлаш учун дастурни компиляция этишда зарур кутубхона модулларини улаш зарур. Буни буйруқ сифатида улаш ёки кўпгина системаларда махсус mpicc (Си тилидаги дастурлар учун), mpicc (Си++ тилидаги дастурлар учун) ва mpif77/mpif90 (Фортран77/90 тилидаги дастурлар учун) буйруқ ва функциялар мавжуд. Компиляторнинг “-o name” бўлими яратиладиган ишчи файл a.out деб номланади, Масалан: mpiff77-o program program.f

Ишчи файл яратилгандан кейин уни зарур миқдордаги жараёнлар учун ишга

тушириш керак. Бунинг учун одатда MPI-дастурларини ишга туширувчи mpirun буйруғи мавжуд, Масалан: mpirun – np N< дастур ва аргументлари>
Бу ерда N – битта масаладаги жараёнлар миқдори. Ишга туширилгандан кейин битта дастур ишга туширилган жараёнлар томонидан чақирилади. Чақириш натижаси системага боғлиқ равишда терминалга ёки номи кўрсатилган файлга ёзилади. Қолган барча объектлар: процедуралар, константалар ва MPI да аниқланган маълумотлар тоифалари MPI олд қўшимчага эга бўлади. Агар фойдаланувчи дастурда бундай қўшимчали номлардан фойдаланмаса хам MPI объектлар билан муаммо туғилмайди. Бундан ташқари Си тилида функция номларидан бош ва кичик харфлар фарқланади. Одатда MPI функция номидаги MPI қўшимчасидан кейинги харф бош харф билан, кейингилари кичик харфлар билан ёзилади. MPI константалар номи эса бутунлигича бош харфлар билан ёзилади. MPI интерфейсининг тавсифи mpif.h (mpi.h) файлида мужассамлаштирган. Шунинг учун MPI – дастур бошқа include ’mpif.h’ директиваси (кўрсатмаси) (Си тилида дастурда #include ’mpi.h’) жойлашиши керак. MPI дастур – бу ўзаро мулоқат қилувчи параллел дастурлар тўпламидир. Ҳар бир жараён бир марта юз беради ва дастурни параллел қисмини ташкил этади. MPI дастур бажарилиши жараёнида қўшимча жараёнларни ташкил этиш ёки мавжудларини йўқотиш мумкин эмас. Ҳар жараён ўзининг манзил сохасида жойлашади ва умумий ўзгарувчилар MPI да йўқ. Жараёнлар ўртасида мулоқатнинг ягона усули хабарлар алмашишдир. Параллел жараёнларни мулоқатини мустақил амалга ошириш учун жараёнлар гурухи танлаб олиниб улар учун алохида мухит коммуникаторлар яратилади. Жараёнлар фақат битта коммуникаторлар ичида мулоқат қилинади ва турли коммуникаторларга юборилган хабарлар кесишмайди. Фотран тилидаги коммуникаторлар INTEGER тоифасига эга бўлади. Дастур ишга тушишида юзага келадиган жараёнлар тўла қамровли MPI_COMM_WORLD номли

коммуникатор доирасида ишлайди деб хисобланади. Ушбу коммуникатор доимо мавжуд бўлади ва ишга тушган барча MPI дастурларни мулоқат қилиш учун хизмат қилади. Бундан ташқари дастур ишга тушишида битта жараёнга эга бўлган MPI_COMM_SELF коммуникатори, хамда битта хам жараёнга эга бўлмаган MPI_COMM_NULL коммуникатори мавжуд бўлади. Жараёнлар ўртасида мулоқатлар маълум коммуниқаторлар доирасида амалга ошади. Турли коммуникаторларга берилган хабарлар кесишмайди. MPI дастуридаги хар бир жараён ўзи тегишли бўлган гурух ичида такрорланмас атрибутга – мусбат бутун сондан иборат жараён номерига эга бўлади. Ушбу атрибут ёрдамида жараёнлар ўртасидаги асосий мулоқатлар олиб борилади. Битта коммуникатордаги жараёнлар хар хил номерга эга бўлади, бироқ жараён бир вақтда бир неча коммуниқаторга тегишли бўлиши мумкин. Бу холда жараённинг битта коммуникатор ичида номери иккинчи коммуникатор ичидаги номеридан фарқ қилиши мумкин. Бундай жараённинг иккита асосий атрибути тушинарли бўлади: коммуникатор ва коммуникаторда номер. Агар гурухда n та жараён бўлса, у холда жорий гурухдаги ихтиёрий жараён номери 0 дан n – 1 гача ораликда ётади. Жараёнларни ўзаро мулоқат қилишининг асосий усули хабарлар алмашишдан иборат. Хабар–бу бирор тоифадаги маълумотлар тўпламидир. Хар бир хабар бир қанча атрибутларга эга бўлади. Масалан, жўнатувчи жараён номери, қабул қилувчи жараён номери, хабар идентификатори ва бошқалар. Хабарнинг асосий атрибутларидан бири унинг идентификатори ёки тэги хисобланади. Хабарни қабул қилаётган жараён хабар идентификатори бўйича унга битта жараёндан келган иккита хабарни фарқлаб олади. Хабар идентификатори мусбат бутун сон бўлиб 0 ва MPI_TAG_UP диопазонда ётади ва MPI_TAG_UP 32767 дан кичик эмас. Хабар атрибутлари билан ишлаш учун массив (Си тилида структура) киритилган. Массив элементлари ёрдамида хабар қийматларига мурожат этилади. MPI процедураларини кўпчилигида охирги аргументда

муваффақиятли тугаганлик хақидаги маълумот қайтарилади. Муваффақиятли бажарилганда MPI_SUCCES қиймати, акс холда хатолик коди қайтарилади. Процедура бажарилишида юз берган хатоликни унинг тавсифидан билиб олиши мумкин. Турли хатолик кодларига мос тавсифлар mpif.h файлида жойлашади.

2.1. MPI технологияси

Дстурни параллел кисмини ўрнатиш MPI_INIT(IERR). Қолган MPI працедуралар MPI_INIT чақирилгандан кейин чақирилиши мумкун. Хар бир дастурни параллел қисмини ўрнатиш фақат бир марта бажарилади Си тилида MPI_INIT функциясида кўрсатгичлар ёрдамида дастурнинг буйруқ сатридаги argc ва argv параметрлари берилади. Улар ёрдамида праралел жараёнларга параметрлар берилади.

Дастурнинг параллел қисми ишини тугатиш MPI_FINALIZE(IERR). MPI нинг ихтиёрий процедураси, шунингдек MPI_INIT га кейин мурожат қилиш тақиқланади. MPI_FINALIZE чақирилишигача хабарлар алмашишда уни иштирок этишини талаб қилувчи дастурнинг барча жараёнлар иши тугатилиши лозим. Фортран тилидаги оддий MPI дастурга мисол:

```
program examplel
include 'mpi.h'
integer ierr
print *,'Before MPI_INIT'
call MPI_INIT(IERR)
print*,'Parallel section'
call MPI_FINALIZE(ierr)
print*,'After MPI_FINALIZE'
end
```

MPI ни амалга ошишига боғлиқ равишда 'Before MPI_INIT' ва 'After MPI_FINALIZE' сатрларини дастурнинг битта жараёни ёки ишга туширилган барча жараёнлар чоп қилиниши мумкин. 'Parallel section' сатрини барча жараёнлар чоп қилади. Турли жараёнлар томонидан сатрни чиқариш тартиби ихтиёрий амалга ошади.

Си тилидаги MPI дастурнинг умумий схемаси қуйидагича тасвирланади:

```
#include "mpi.h"
main(int argc,char **argv)
{
   ...
  MPI_INIT(&argc,&argv);
  ...
   MPI_FINALIZE();
   ...
}
```

```
MPI_INITIALIZED(FLAG,IERR)
LOGICAL FLAG
INTEGER IERR
```

Процедура FLAG аргументида агар процедура дастурнинг параллел қисмидан чақирилса TRUE, акс холда FALSE қийматини қайтаради. Навбатдаги процедура MPI_INIT чақирилишигача чақирилиши мумкин бўлган ягона MPI процедурадир:

```
MPI_COMM_SIZE(COMM,SIZE,IERR)
INTEGER COMM,SIZE,IERR
```

Процедура SIZE аргументида COMM каммуникаторидаги параллел жараёнлар сонини қайтаради.

```
MPI_COMM_RANK(COMM,RANK,IERR)
```

Процедура RANK аргументида COMM коммуникаторидаги жараён номерини қайтаради. Агар MPI_COMM_SIZE процедураси COMM коммуникатори учун SIZE қийматини қайтарса, у холда MPI_COMM_RANK процедурасининг RANK ўзгарувчи қиймати 0, size–1 диопазонда бўлади.

Навбатдаги мисолда хар бир ишга туширилган жараён ўзининг MPI_COMM_WORLD коммуникаторидаги номерини ва жорий

коммуникатордаги жараёнлар сонини қайтаради.

```
program example2
include 'mpi.h'
integer   ierr,size,rank
call MPI_INIT(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD, size,ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, rank,ierr)
print  *, 'process',rank,size,size
call MPI_FINALIZE(ierr)
end
```

Print процедураси чақирилиши натижасида тасвирланувчи сарт дастур ишга тушишида қанча жараён яратилган бўлса шунча марта тасвирланади, сатрларни чиқариш тартиби олдиндан маълум эмас ва ихтиёрий тартибда бўлиши мумкин. Бироқ хар бир тасвирланувчи сатр бир-бири билан аралашиб кетмаслиги кафолатланади.

```
DOUBLE PRECISION MPI_WTIME(ierr)
INTEGER IERR
```

Бу функция чақирувчи жараёнда ўтган бирор вақт моментидан жорий вақтгача ўтган вақтни секундларда қайтаради. Агар дастурнинг бирор қисми ушбу функция ичига олинса, у холда қийматлар фарқи жорий қисмни бажарилиш вақтини кўрсатади. Бунда ўтган моменти ўзгармайди. Ушбу функция ўз ишини натижасини параметр билан эмас, очиқ холда қайтаради. Турли жараёнларнинг таймерлари синхронлашмаган бўлиши мумкин ва турлича қиймат қайтариш мумкин. Бунда MPI_WTIME_IS_GLOBAL (1-синхронлашган ,0-йўқ) парамер қиймати орқали аниқлаш мумкин.

```
DOUBLE PRECISION  MPI_WTICK(IERR)
INTEGER  IERR
```

Бу функция чақирувчи жараёнда таймерни қийматини секундларда

қайтаради. Бу функция хам ўз иш натижасини параметр орқали эмас, балки очиқ холда қайтаради.

MPI_GET_PROCESSOR_NAME(NAME ,LEN,IERR)

CHARACTER *(*) NAME

INTEGER LEN , IERR

Бу функция NAME сатрида чақирувчи жараён ишига тушурилган тугун номини қайтаради. LEN ўзгарувчида номдаги символлар миқдори қайтарилади. Бу қиймат MPI_GET_PROCESSOR_NAME константа қийматидан ошмайди. Бу процедура ёрдамида MPI дастурнинг жараёнлари қайси физик процессорга мўлжалланганлиги аниқланади.

Навбатдаги мисолда MPI_GET_PROCESSOR_NAME процедурасини қўллаш кўрсатилган:

```
program example3
include 'mpi.h'
integer ierr,rank,len,i,NTIMES
parametr (NTIMES=100)
character*(MPI_MAX_PROCESSOR_NAME) name
double precision time_start,time_finish,tick
call MPI_INIT(IERR)
call MPI_COMM_RANK(MPI_COMM_WORLD,rank, ierr)
call MPI_GET_PROCESSOR_NAME(name,len,ierr)
tick = MPI_WTICK(ierr)
time_start = MPI_WTIME(ierr)
do  i=1 , NTIMES
time_finish = MPI_WTIME(ierr)
end do
call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
print *,'processor',name(1:len),
```

```
&   'process',rank,': tick =', tick,
&   'time=',(time_finish-time_start)/NTIMES
call MPI_FINALIZE(ierr)
end
```

2.2.Алохида жараёнлар ўртасидаги хабарларни узатиш-қабул қилиш

Амалда MPI коммуникацион технологиясидан фойдаланиб ёзилган барча дастурлар фақат параллел жараёнларни яратиш ва тугатиш воситаларига эмас, балки жараёнлар ўзаро мулоқат қилиш воситасига хам эга бўлиши керак. MPI да бундай мулоқат хабарларни яққол узатиш орқали амалга оширилади.

MPI даги хабарлар алмашуви процедуралар иккита гурухга бўлинади.

Бирчи гурухга дастурнинг фақат иккита жараёни мулоқат қилиш учун мўлжалланган процедуралар киради. Бу амал индивидуал ёки нуқта-нуқта деб аталади.

Иккинчисига гурух процедураларида коммуникатордаги барча жараёнлар хабарлар алмашиши инобатга олинади. Бундай амаллар жамоа деб аталади.

Хабарлар алмашиш процедураларини тахлил қилиш нуқта-нуқта тоифали амал бажарувчи процедуралардан бошлаймиз. бундай мулоқатда иккита жараён қатнашади ва уларнинг бири жўнатувчиси, иккинчиси қабул қилувчи хисобланади. Жўнатувчи жараён маълумотлар алмашишини таъминловчи процедуралардан бирини чиқариши керак ва бирор коммуникатродаги қабул қилувчи жараён номерини кўрсатиши керак. қабул қилувчи жараён эса узатувчи жараён номерни кўрсатган холда қабул қилиш процедурасидан бирини чиқариши керак. баъзи холларда узатувчи жараён номери номаълум бўлиши мумкин. жорий гурух процедуралари хам ўз навбатда иккита синфга бўлинади: блокировкали процедура (синхронли) ва

блокировкасиз процедура (асинхрон).

Блокировкали хабар олиш процедуралари алмашиши бирор шарт бажарилишигача ўзи ишни тугатади. Асинхрон процедураларини қайтиши эса мос коммуникатцион амал бажарилиши дарров амалга ошади.

Блокировкали процедурадан тўғри фойдаланишдан боши берг холатлар вужудга келиши мумкин. Шунинг учун қўшимча эхтиёт чораларни кўриш керак.

Асинхрон процедураларидан фойдаланишда боши берк холатлар юз бермайди, бироқ уларни қўллашда маълумотлар массивидан тўғри фойдаланиш талаб этилади.

MPI_SEND(BUF,COUNT,DATATYPE,DEST,MSGTAG,COMM,IERR)

<type> BUF(*)

INTEGER COUNT,DATATYPE, DEST,MSGTAG ,COMM, IERR

Блокировкали массив жўнатиш DATATYPE тоифали COUNT та элементлардан иборат MSGTAG идентификаторли BUF хабари COMM коммуникаторидаги DEST номерли жараёнига жўнатилган. Жўнатилган хабарларнинг барча элеиентлари кетма-кет холда BUF буферда жойлашиши керак. Бу амал қабул қилувчи жараён ўрнатилмаган бўлса хам бажарилади. Бунда хабар бевосита қабул буферига нусхалангани каби бирор система буферига хам жойлаштирилади (агар MPI да белгиланган бўлса). COUNT қиймати ноль бўлиши хам мумкин. Жараён ўз-ўзига хам хабар бериши мумкин. Бу хавфли эмас ва боши берг холатларга хам, олиб келиши мумкин. DATATYPE параметри Фотран тилида INTEGER тоифасига эга бўлади(СИ тилида MPI_Datatype). Узатиладиган элементлар тоифаси Фотран тилидан константалар билан кўрсатилади. Бу константалар қуйидаги жадвалда келтирилган:

MPI ДАГИ МАЪЛУМОТЛАР ТОИФАСИ	ФОТРАНДАГИ МАЪЛУМОТЛАР ТОИФАСИ
MPI_INTEGER	integer
MPI_REAL	REAL
MPI_DOUBLE_PRECISION	DOUBLE PRECISION
MPI_COMPLEX	COMPLEX
MPI_LOGICAL	LOGICAL
MPI_CHARACTER	CHARACTER(1)
MPI_BYTE.	8 бит, тоифалашмаган маълумотлар учун қўлланилади.
MPI_PACKED	Маълумотлар жойлашиш тоифаси.

Агар MPI қўлланиладиган тил маълумотларининг қўшимча тоифасига эга бўлса, у холда бу тоифалар mpi да хам қўллаб қувватланиши керак. Мавжуд тоифаларнинг тўлиқ рўйхати mpif.h(mpi.h) файлида келтирилади.

Хабарлар жўнатишда мавжуд бўлмаган жараёнлар учун махсус MPI_PROC_NULL қийматини қўллаш мумкин. Бундай жараёнли операциялар MPI_SUCCESS тугатиш коди билан тез тугайди. Масалан, бир бирликка ортиқ номерли жараёнга хабар жўнатиш учун қуйидаги фрагментдан фойдаланиш мумкин:

```
call MPI_COMM_SIZE(MPI_COMM_WORLD, size, ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
next = rank+1
if(next .eq. size) next =MPI_PROC_NULL
call MPI_SEND(buf,1,MPI_REAL,next,
```

& 5, MPI_COMM_WORLD,ierr)

Бу холда жараён хақиқатда маълумотлар жўнатмайди ва дастурни бажаришда давом этади.

Блокировка процедурадан қайтгандан кейин барча параметрларни такрорий қўлланилишини кафолатлайди. Яъни mpi_send дан қайтгандан кейин жорий процедурани чақиришда қўлланилган ўзгарувчиларнинг ихтиёрий бирини берилган хабарни бузилишга хавфсирамасдан ишлатиш мумкин. Буни кафолатлаш усулини танлаш: оралиқ буферга нусхалаш ёки бевосита dest жараёнига бериш mpi ни яратаётган дастурчилар ихтиётига боғлиқ.

Шуни таъкидлаб ўтиш керак, mpi_send жараёнидан қайтиш хабар dest жараённинг жараён элементини тарк этганини хам билдирмайди. Фақат жорий процедурани чақиришда фойдаланилган ўзгарувчиларни бехатар ўзгартириш кафолатланади. Шунга ўхшаш ноаниқликлар хар доим хам фойдаланувчиларни қаноатлантирмайди. Хабарлар алмашиш имкониятини кенгайтириш учун mpi да учта процедура мавжуд. Бу процедуралардаги параметрлар хам худди mpi_send даги каби, бироқ уларни хар бири ўзига хос хусусиятга эга.

MPI қуйидаги маълумотлар алмашиш процедуралар модификацияларини қўллаб қувватлайди:

- MPI_BSEND буфератция билан хабар узатиш. Агар узатилган хабар қабул қилувчи томонидан қабул қилинмаса, у холда хабар махсус буферга ёзилади ва тезда процедурадан қайтарилади. Бу процедурани бажарилиши мос қабул қилувчи процедурани чақирилишига боғлиқ эмас, қолаверса, агар буферда етарлича жой бўлмаса процедура хатолик кодини қайтаради. Буферлаш учун массив белгилашни ишини фойдаланувчи ўз зиммасига олиши керак.

- MPI_SSEND синхронлаб хабарлар узатиш. Ушбу процедурадан чиқиш

фақат жўнатилган хабар қабул қилувчи жараён томонидан қабул қилингандан кейин мумкин. Бу билан хабар алмашишни синхронлаб тугатиш орқали жўнатма буферидан такрорий фойдаланиш имконияти яратилади ва дастурдаги қабул қилувчи жараён хабарни қабул қилиши кафолатланади. Синхронлаб хабар алмашиш дастур бажарилишини секинлатади, бироқ системада хабарларни ортиқча буферлардан холи қилади.

-MPI_RSEND таёргарлик бўйича хабар узатиш. Бу процедурадан фақат қабул қилувчи жараён қабулни амалга оширгандан кейин фойдаланиш мумкин. Акс холда бу процедура чақирилиши нотўғри хисобланади ва унинг натижаси мавхум бўлади. Mpi_rsend процедураси чақирилишидан олдин хабар қабул қилинганлигини кафолатлаш учун жараёнларни яққол ва нояққол синхронлаш оперцияларидан фойдаланиш керак. Кўпинча mpi_rsend процедураси узатувчи ва қабул қилувчи ўртасидаги мулоқат баённомасини қисқартиради ва маълумотлар алмашишдаги сарф-харажатларни камайтиради.

Фойдаланувчи жўнатувчи жараёнда mpi_bsend процедурасини чақирганда хабарларни буферлаш учун зарур махсус массив белгилаш керак.

MPI_BUFFER_ATTACH(BUF,SIZE,IERR)

<type> BUF(*)

INTEGER SIZE, IERR

SIZE ўлчами BUF массивини буферлаб хабар жўнатишда ишлатиш учун яратиш. Бундай буфердан хар бир жараёнда фақат биттаси бўлиши керак. Буфер сифатида белгиланган массивдан дастурда мақсадларда ишлатмаслик керак. Буферлаш учун яратилган массив ўлчами умумий ўлчамидан кичик бўлмаслиги керак ва минимум MPI_BSEND_OVERHEAD константасидан аниқланган қийматга тенг бўлиши керак.

MPI_BUFFER_DETACH(BUF, SIZE, IERR)

<type> BUF (*)

INTEGER SIZE, IERR

Ажралган буфер массивини бошқа мақсадларда ишлатиш учун бўшатиш. Бу процедура BUF ва SIZE аргументларида бўшатилган массив манзили ва ўлчами қийматларини қайтаради. Бу процедурани чақирган жараён жорий буфердаги хабарлар жўнатилгунча уни блокировкалаб туради. Одтда MPI да жўнатиладиган хабарларни буферлаш учун бирор ўлчамдаги хотира қўлланилади. Бироқ дастур барча буферловчи жўнатмаларга етарли буферни аниқ кўрсатиш тавсия этилади.

Навбатдаги мисолда буферлаб хабарлар алмашиш қўлланилган:

```
program example4
include 'mpi.h'
integer BUFSIZE
parameter (BUFSIZE =4+MPI_BSEND_OVERHEAD)
byte buf (BUFSIZE)
integer rank,ierr,ibufsize, rbuf
integer status(MPI_STATUS_SIZE)
call MPI_INIT(ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)
if(rank .eq.  0)  then
call  MPI_BUFFER_ATTACH(buf,BUFSIZE, ierr)
call  MPI_BSEND(rank,1,MPI_INTEGER,1, 5, MPI_COMM_WORLD,ierr)
&  call MPI_BUFFER_DETACH(buf,ibufsize,ierr)
end if
if(rank  .eq. 1) then
call MPI_RECV(rbuf,1,MPI_INTEGER,0,5,MPI_COMM_WORLD,ierr)
print  *,'Process 1 received',rbuf,'from process',status(MPI_SOURCE)
end if
call MPI_FINALIZE(ierr)
```

end

MPI_RECV(BUF,COUNT,DATATYPE,SOURCE,MSGTAG,COMM,
STATUS,IERR)

<type> BUF(*)

INTEGER COUNT,DATATYPE,SOURCE,MSGTAG,COMM,IERR,
STATUS(MPI_STATUS_SIZE)

DATATYPE тоифали, элементлари count дан кўп бўлмаган MSGTAG идентификаторли хабар элементлари билан buf буферига блокировкалаб қабул қилиш. Агар реал қилинган элементлар сони count миқторидан кам бўлса, у холда BUF буферида қабул қилинган хабар элементларга мос элементлар ўзгаради. Агар қабул қилинаётган элементлари count қийматидан кўп бўлса, у холда хатолик юз беради. Буни бартараф этиш учун MPI_PROBE(MPI_IPROBE) процедураси ёрдамида келадиган хабар структураси аниқланади. Агар қабул қилинган хабарларда элементлар сонини аниқ билиш керак бўлса, у холда MPI_GET_COUNT процедурасидан фойдаланиш мумкин. Блокировкалаш амали MPI_RECV процедурасидан қайтгандан кейин хабарнинг барча элементлари қабул қилинган ва buf буферига жойлашган бўлади.

Қуйидаги келтирилган дастурда ноль жараёни бир рақамли жараёнга хабар жўнатади ва ундан жавоб кутади. Агар дастур кўп сонли жараён билан ишга туширилса, у холда жўнатмалар бажариш ноль бўлади. Қолган жараёнлар MPI_INIT процедураси ёрдамида ташкил этилгандан кейин a ва b ўзгарувчиларнинг бошланғич қийматларини чоп қилади, кейин MPI_FINALIZE процедурасини бажариб ишини тугатади.

```
program example5
include 'mpi.h'
integer ierr,size,rank
real a,b
```

```
integer status(MPI_STATUS_SIZE)
call MPI_INIT(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD,size,ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)
a=  0.0
b= 0.0
if(rank  .eq. 0) then
b=1.0
call MPI_SEND(b,1,MPI_REAL,1 ,5,MPI_COMM_WORLD,ierr);
call MPI_RECV(a,1,MPI_REAL, 1,5,MPI_COMM_WORLD,status,ierr);
else
if (rank  .eq. 1) then
a=2.0
call MPI_RECV(b,1,MPI_REAL, 0,5,MPI_COMM_WORLD,status,ierr);
call MPI_SEND(a,1,MPI_  REAL ,0,5,MPI_COMM_WORLD,ierr);
end if
end if
print  *,'process',rank,'a=',a,'b=',b
call MPI_FINALIZE(ierr)
end
```

Навбатдаги мисолда хар бир тоқ номерни жараён бир бирликка, кўп номерли қўшни жараёнга хабар жўнатади. Қўшимча равишда энг катта номерни жараёнга хабар жўнатмаслиги учун текширилади. b ўзгарувчининг қиймати фақат тоқ номерли жараёнларда ўзгаради.

```
program example6
include 'mpif.h'
integer ierr,size,rank,a,b
integer status(MPI_STATUS_SIZE)
```

```
call MPI_INIT(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD,size,ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)
a=rank
b=-1
if(mod(rank,2) .eq. 0) then
if(rank+1 .lt. size) then
call MPI_Send(a,1,MPI_INTEGER,rank+1,5,MPI_COMM_WORLD,ierr);
end if
else
callMPI_Recv(b,1,MPI_INTEGER,rank-1,5,MPI_COMM_WORLD,status,ierr);
end if
print *,'process',rank,'a=',a,'b=',b
call MPI_FINALIZE(IERR)
end
```

Келтирилган мисолда SOURCE ва MSGTAG аргументлари ўрнига қуйидаги константаларни қўллаш мумкин:

- MPI_ANY_SOURCE итиёрий жараёндан берилган хабар мос келишини билдирувчи белги;
- MPI_ANY_TAG ихтиётий идентификаторли хабар мос келишини билдирувчи белги;

Ушбу иккита константа баравар ишлатилганида ихтиёрий жараённинг ихтиёрий идентификаторли хабар қабул қилинади. Қабул қилинган хабарнинг аниқ атрибутларини status массивининг мос элементлари ёрдамила аниқлаш мумкин. Фотран status параметри бутун сонли ва MPI_STATUS_SIZE ўлчами массив хисобланади. MPI_SOURCE, MPI_TAG ва MPI_ERROR константалари берилган массивнинг мос элементлари қийматларига мурожат этиш қоидаси хисобланади:

- status (MPI_SOURCE) хабарни жўнатувчи жараён номери;
- status (MPI_TAG) хабар идентификатори;
- status (MPI_ERROR) хатолик коди;

Си тилидаги status параметри MPI_SOURCE, MPI_TAG ва MPI_ERROR майдонларидан иборат MPI_SOURCE тоифали структура хисобланади.

Хабарларни жўнатиш ва қабул қилиш амалларининг баъзи носимметрик жихатларига эътибор қилинг. MPI_ANY_SOURCE константа ёрдамида ихтиёрий жараёндан хабар қабул қилиш мумукин. Бироқ маълумотларни жўнатишда қабул қилувчи жараённинг номерини аниқ кўрсатиш керак. Стандартга кўра, агар бир жараён битта MPI_RECV чақирувга кетма-кет иккита хабар жўнатса, иккинчи жараён дастлаб олдин жўнатилган хабарни қабул қилади. Шу билан бирга агар иккита хабар бир вақтда турли жараёнлар томонидан жўнатилса у холда қабул қилувчи жараён томонидан хабарларни қабул қилиш тартиби олдиндан маълум эмас.

MPI_GET_COUNT(STATUS,DATATYPE,COUNT,IERR)

INTEGER COUNT,DATATYPE,IERR,STATUS(MPI_STATUS_SIZE)

status параметрининг қийматлари орқали процедура қабул қилинган хабарнинг (MPI_RECV га мурожат этгандан кейин) ёки DATATYPE тоифали хабарнинг қабул қилинган элементларининг (MPI_PROBE ёки MPI_IPROBE га мурожат этиб) COUNT миқдорини аниқлайди. Жорий процедура қабул қилинган хабарларни сақлаш учун ажратилган хотира сохасининг хажмини ўлчаш учун зарур.

MPI_PROBE(SOURCE,MSGTAG,COMM,STATUS,IERR)

INTEGER SOURCE,MSGTAG,COMM,IERR,STATUS(MPI_STATUS_SIZE)

status массивида COMM коммуникаторидаги SOURCE номери жараёндан кутилаётган MSGTAG идентификатори хабар структураси хақидаги маълумотларни олиш. Бу процедурадан қайтиш мос идентификаторли хабар ва мос жўнатувчи жараён номери ўқиш учун мумкин бўлгандагина амалга

ошади. Бу процедура хабар келиш фактини аниқлайди (уни қабул қилмайди). MPI_PROBE чақирилгандан кейин MPI_RECV хам шу параметрлар билан чақирилса, у холда хам MPI_PROBE процедураси ёрдамида олинган хабарлар қабул қилинади.

Навбатдаги мисолда келувчи хабар структурасини аниқлаш учун MPI_PROBE процедурасини қўллаш намоиш этилган. 0 жараёни 1 ва 2 жараёнларнинг ихтиёрий биридан хабар кутади. Бироқ ушбу жараёнлар жўнатадиган хабарлар хар хил тоифага эга. Келувчи хабарни қайси ўзгарувчига жойлашишни аниқлаш учун жараён аввал MPI_PROBEчақируви ёрдамида хабар кимдан келганлигини аниқлайди. Бевосита MPI_PROBE дан кейинги MPI_RECV чақируви ишончли равишда хабарни қабул қилади ва шундан кейин бошқа жараёндан хабар қабул қилинади.

```
program example7
include 'mpif.h'
integer rank,ierr,ibuf,status(MPI_STATUS_SIZE)
real rbuf
call MPI_INIT(ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)
ibuf=rank
rbuf=1.0*rank
if(rank.eq.1)
call MPI_SEND(ibuf,1,MPI_INTEGER,0,5,MPI_COMM_WORLD,ierr)
if(rank.eq.2)
call MPI_SEND(rbuf,1,MPI_REAL, 0,5,MPI_COMM_WORLD,ierr)
if(rank .eq.0) then
call MPI_PROBE(MPI_ANY_SOURCE,5,MPI_COMM_WORLD,ierr)
if(status(MPI_SOURCE) .EQ.1) then
call MPI_RECV(ibuf,1,MPI_INTEGER,1,5,MPI_COMM_WORLD,status,ierr)
```

```
call MPI_RECV(rbuf,1,MPI_REAL,2,5,MPI_COMM_WORLD,status,ierr)
else
if(status(MPI_SOURCE) .EQ.2) then call
MPI_RECV(rbuf,1,MPI_REAL,2,5,MPI_COMM_WORLD,status,ierr)
call MPI_RECV(ibuf,1,MPI_INTEGER,1,5,MPI_COMM _WORLD,status,ierr)
end if
end if
print *,'Process 0 recv' ,ibuf,'from process 1',rbuf,'from process 2'
end if
call MPI_FINALIZE(ierr)
end
```

Навбатдаги мисолда иккита жараённи галма-гал хабар алмашиш модели кўрсатилади. Бунда хабар алмашиш вақти хабар узунлигига боғлиқ аниқланади. Шу тарзда параллел компьютердаги коммуникатцион тармоқнинг асосий характеристикаси аниқланади. Булар: яширинганлик (ноль узунликдаги хабарни алмашиш вақти) ва максимал ўтказувчанлик қобиляти (бир секундаги Мбайт миқдорида), ҳамда хабарнинг узунлиги, NMAX константа узатиладиган хабарнинг максимал узунлиги чекланишини билдиради, NTIMES константаси эса натижани ўртачасини аниқлашдаги такрорланишлар миқдорини билдиради. Яширинликни аниқлаш учун аввал ноль узунликдаги хабар жўнатилади, кейин хабар узунлиги икки марта кўпаяди. Бунда жўнатиш аввал real*8 тоифали битта элементни жўнатишдан бошланади.

```
program example8
include 'mpif.h'
integer ierr,rank,size,i,n,1max,NMAX,NTIMES
parameter (NMAX=1 000 000,NTIMES=10)
double precision time_start,time,bandwidth,max
```

```
real *8 a(NMAX)
integer status (MPI_STATUS_SIZE)
call MPI_INIT(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD,size,ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)
time_start=MPI_WTIME(ierr)
n=0
max= 0.0
lmax=0
do while (n.le.NMAX)
  time_start=MPI_WTIME(ierr)
do i=1,NTIMES
     if(rank .eq. 0) then
callMPI_SEND(a,n,MPI_DOUBLE_PRECISION,1,1,
MPI_COMM_WORLD,ierr)
callMPI_RECV(a,n,MPI_DOUBLE_PRECISION,1,1,
MPI_COMM_WORLD,status,ierr)
end if
if(rank .eq. 1) then
callMPI_RECV(a,n,MPI_DOUBLE_PRECISION,0,1,
MPI_COMM_WORLD,status,ierr)
callMPI_SEND(a,n,MPI_DOUBLE_PRECISION,0,1,
MPI_COMM_WORLD,ierr)
end if
enddo
time=(MPI_WTIME(ierr)-time_start)/2/NTIMES
bandwidth=(8*n*1.d0/(2**20))/time
if(max .lt. bandwidth) then
```

```
max=bandwidth
lmax=8*n
end if
if(rank .eq.0) then
 if(n.eq.0) then
print *,'latency=',time,'seconds'
else
print *, 8*n,'bytes,bandwidth=',bandwidth,'Mb/s'
end if
end if
if(n .eq.0) then
n=1
else
n=2*n
end if
end do
if(rank .eq.0) then
print *, 'max bandwidth=', max, 'Mb/s,length='
,lmax,'bytes'
end if
call MPI_FINALIZE(ierr)
end
```

Блокировкаланган хабарларни жўнатиш ва қабул қилиш.

MPI да маълумотларни асинхрон алмашиш учун бир қатор процедуралар мавжуд. Блокировкаловчи процедуралардан фарқли равишда бундай процедуралардан қайтиш чақирилган кейин хеч бир жараённи иши тўхтатилмасдан дарров амалга ошади. Дастур бажарилиши давомида асинхрон ишга туширилган амаллар юз беради ва қайта ишланади. Амалда,

бу имконият эффектив дастурлар тузишда жуда фойдали хисобланади. Аслида, дастурчи бирор холатда унга бошқа жараёнда хисобланаётган массив керак бўлишини билади. Дастурчи олдиндан жорий массивни олиш учун дастурда асинхрон сўров жойлайди. Массив хақиқатда зарур бўлишига қадар у бошқа бир фойдали мақсадларда қўлланилиб туради. Бу ерда хам кўпчилик холатларда хабарни жўнатиб бўлинишини кутмасдан навбатдаги хисоблашларни бажариш мумкин. Асинхрон хабар алмашишни тугатиш учун хабар алмашиш амали тугаганлигини ёки давом этаётганлигини текширувчи қўшимча процедура зарур бўлади. Фақат шундан кейингина алмашинаётган хабарни бузилишига хавфсирамасдан жўнатиш буферидан фойдаланиш мумкин. Агар хабарларни узатиш қабул қилиш амалларини хисоблашлар фонида яширин имконияти мавжуд бўлса, ундан фойдаланиш керак. Бироқ амалиёт доимо назария билан мос келавермайди. Бу конкрет хақиқийликка боғлиқ. Афсуски, асинхрон амаллар аппаратура ва система томонидан эффектив қўллаб-қувватланмайди. Шунинг учун фонда хисоблашларни бажариш эффекти нольга тенг ёки унчалик катта бўлмаса хайрон бўлмаслик керак. Келтирилган эслатмалар эффективлик билан боғлиқ. Шундай бўлсада асинхрон амаллар функционаллиги бўйича жуда фойдали хисобланади ва шунинг учун асинхрон амаллар хар бир хақиқий дастурда мавжуд.

MPI_ISEND(BUF,COUNT,DATATYPE,DEST,MSGTAG,COMM,
REQUEST,IERR)
<type> BUF(*)
INTEGER COUNT,DATATYPE,DEST,MSGTAG,COMM,REQUEST,IERR

BUF буферидан блокировкаламасдан COMM коммуникаторидаги DEST жараёнига MSGTAG идентификаторли DATATYPE тоифаси хабарнинг COUNT элементларини жўнатиш. Бу процедурадан қайтиш узатиш жараёни яратилгандан кейин, BUF буферидаги тўла хабар қайта ишланиб бўлинмасидан дархол амалга ошади. Яъни бу холат жорий буфердан

жўнатиш ишлари тугатилганлигини тасдиқловчи қўшимча маълумотларни олмасдан фойдаланиш мумкин эмаслигини билдиради. BUF буферидан ундаги узатилаётган хабарни бузилиш хатаридан холи равишда қайта фойдаланиш моментини қайтарувчи REQUEST параметри ёрдамида MPI_WAIT хамда MPI_TEST тоифали процедуралар ёрдамида аниқлаш мумкин. REQUEST параметри Фотран тилида INTEGER тоифага эга ва маълум блокировкаланмаган амални идентификатциялаш учун қўлланилади. MPI_SEND процедурасини учта модификатциясига мос учта қўшимча MPI_ISEND процедура мавжуд. Булар қуйидагилар:

- MPI_IBSEND буфердаги хабарни блокировкаламасдан жўнатиш;
- MPI_ISSEND хабарларни синхронли блокировкаламасдан жўнатиш;
- MPI_IRSEND таёрлиги бўйича блокировкаламасдан хабар жўнатиш;

MPI_IRECV(BUF,COUNT,DATATYPE,SOURCE,MSGTAG,
COMM,REQUEST,IERR) <type> buf(*)
INTEGER COUNT ,DATATYPE,SOURCE,MSGTAG,COMM,REQUEST,IERR

STATUS массивини тўлдирган холда COMM коммуникаторидаги SOURCE номерли жараёндан MSGTAG идентификаторли DATATYPE тоифали хабарни COUNT дан кўп бўлмаган элементларини блокировкаламасдан BUF буферига қабул қилиш. Блокировкалаб қабул қилишдан фарқли равишда процедурадан қайтиш қабул жараёни юзага келиш билан ва тўла хабар қабул қилиниши, хамда у BUF буферидан ёзилмасдан дарров амалга ошади. Қабул жараёни тугаганлиги REQUEST парметри, хамда MPI_WAIT ва MPI_TEST оиласидаги процедуралари ёрдамида аниқланади.

MPI_SEND, MPI_ISEND процедураларини ихтиёрий биридан, уларнинг учта модификатциясини ихтиёрий биридан жўнатилган хабар MPI_RESV ва MPI_IRESV процедураларини ихтиёрий бири томонидан хабар қабул қилинади.

Блокировкаланмаган амалларни қўллашда бу амал тугамагунича ишлатилаётган массивга маълумотлар ёзмаслик керак.

MPI_IPROBE(SOURCE,MSGTAG,COMM,FLAG,STATUS,IERR)

LOGICAL FLAG

INTEGER SOURCE,MSGTAG,COMM,IERR,STATUS(MPI_STATUS_SIZE)

COMM коммуникаторидаги SOURCE номерли жараёндан кутилаётган MSGTAG идентификаторли хабар структураси хақида маълумотларни STATUS массивида жойлаш. Агар мос атрибутли хабар қабул қилиниши мумкин бўлса, FLAG параметри қиймати TRUE га тенг кўрсатилган атрибутли хабарни хали қабул қилиш мумкин бўлмаса FLAG параметри қиймати FALSE га тенг бўлади.

MPI_WAIT(REQUEST,STATUS,IERR)

INTEGER REQUEST,IERR,STATUS(MPI_STATUS_SIZE)

REQUEST идентификатори билан мос боғланган COUT асинхрон амалини тугашини кутиш. Блокировкаланмаган қабул қилиш амаллари учун STATUSES массивида мос параметр аниқланади. Агар битта ёки бир неча маълумотлар алмашиш амалида хатолик юз берса, у холда STATUSES массивидаги элементларнинг хатолик майдонида мос қийматлар ўрнатилади. Процедура бажарилгандан кейин REQUEST параметриниг мос элементлари MPI_REQUEST_NULL қийматида ўрнатилади.

Қуйидаги келтирилган мисолда барча жараёнлар блокировкаланмаган амаллар ёрдамида халқа топологиясига мос холда яқин қўшнилари билан хабарлар алмашади. Ушбу мақсадда блокировкаловчи амаллар қўлланилса тупик холатлар юзага келади.

```
program example9
include 'mpif.h'
integer ierr,rank,size,prev,next,reqs(4),buf(2)
integer stats(MPI_STATUS_SIZE,4)
```

```
call MPI_INIT(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD,size,ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)
prev=rank-1
next=rank+1
if(rank .eq. 0) prev =size-1
if(rank .eq. size-1) next=0
callMPI_IRECV(buf(1),1,MPI_INTEGER,prev,5,
MPI_COMM_WORLD,reqs(1),ierr)
callMPI_IRECV(buf(2),1,MPI_INTEGER,next,6,MPI_COMM_WORLD,
reqs(2),ierr)
callMPI_ISEND(rank,1,MPI_INTEGER,prev,6,MPI_COMM_WORLD,
reqs(3),ierr)
callMPI_ISEND(rank,1,MPI_INTEGER,next,5,MPI_COMM_WORLD,
reqs(4),ierr)
call MPI_WAITALL(4,reqs,stats,ierr);
print *,'process',rank,'prev=',buf(1),'next=',buf(2)
call MPI_FINALIZE(ierr)
end
```

```
MPI_WAITANY(COUNT,REQUEST,INDEX,STATUS,IERR)
INTEGER COUNT,REQUEST(*),INDEX,STATUS(MPI_STATUS_SIZE),IERR
```

REQUEST идентификатори билан боғланган асинхрон COUNT амаллардан бирини тугашини кутиш. Агар чақирув вақтигача кутилаётган амаллардан бир неча тугаган бўлса, у холда тасодифий равишда ихтиёрий бири танланади. INDEX параметрида REQUEST массивидаги тугаган амал идентификатор сақланувчи элемент номери жойлашади. Блокировкаламайдиган қабул қилишда STATUS параметри аниқланади. Процедура бажарилгандан кейин REQUEST массивининг мос элементида

MPI_REQUEST_NULL қиймати ўрнатилди.

MPI_WAITSOME(INCOUNT,REQUEST,OUTCOUNT,INDEXES,
STATUSES,IERR)
INTEGER INCOUNT,REQUEST(*),
OUTCOUNT,INDEXES(*),IERR,STATUSES(MPI_STATUS_SIZE,*)

REQUEST идентификатори билан боғланган INCOUNT та асинхрон амаллардан бирини тугашини кутиш. INDEXES массивининг дастлабки OUTCOUNT элементида REQUEST массиви элементларининг номери ва идентификатори сақланади. STATUSES массивининг дастлабки OUTCOUNT элементида тугаган амаллар параметрлари сақланади. Процедура бажарилиши тугагандан кейин REQUESTS параметрининг мос элементларида MPI_REQUEST_NULL қиймати ўрнатилади.

Навбатдаги мисолда "master-slave" (барча жараёнлар битта жараён билан мулоқат қилади) коммуникацион схемани ташкил этиш учун MPI_WAITSOME процедурасидан фойдаланиш кўрсатилган. 0 жараёнидан ташқари барча жараёнлар циклнинг хар бир итерациясига slave процедурасини чақириш орқали массивнинг ўзига тегишли локал қисмини аниқлаб олади ва шундан кейин уни бош жараёнга жўнатади. 0 жараёни аввал қолган барча жараёнлардан блокировкаланган қабулни белгилайди, real дан кейин ақалли биттага хабар келишини кутади. Келувчи хабарлар учун 0 жараёни қайта ишловчи master жараёнини чақиради. Бундан кейин яна блокировкаланмаган қабулни ўрнатади. Шу тарзда 0 жараёни жорий вақтда тайёр бўлган маълумотлар блокини қайта ишлайди. Бунда дастур тўғри ишлаши учун 0 жараёни келган хабарларни қайта ишлаб улгуришини таъминлаш керак. Яъни slave процедураси master процедурасига нисбатан кўпроқ ишлаши керак. Бундан ташқари мисолда чексиз цикл ёзилган, шунинг учун аниқ дастур учун ишни тугатиш шарти бўлиши керак.

program example10

```
include 'mpif.h'
integer rank,size,ierr,N,MAXPROC
parametr(N=1000,MAXPROC=128)
integer req(MAXPROC),num,indexes(MAXPROC)
integer statuses(MPI_STATUS_SIZE,MAXPROC)
double precision a(N,MAXPROC)
call MPI_INIT(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD,size,ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)
if(rank .ne. 0) then
do while (.TRUE.)
call slave (a,N)
callMPI_SEND(a,N,MPI_DOUBLE_PRECISION,0,5,
MPI_COMM_WORLD,ierr)
end do
else
do i=1,size-1
callMPI_IRECV(a(l,i),N,
MPI_DOUBLE_PRECISION,i,5,MPI_COMM_WORLD,req(i),ierr)
end do
do while (.TRUE.)
call MPI_WAITSOME(size-1,req,num,indexes,statuses,ierr)
do i=1,num
call master(a(1,indexes(i)),N)
callMPI_IRECV(a(1,indexes(i)), N,MPI_DOUBLE_PRECISION,
indexes(i),5,MPI_COMM_WORLD,req(indexes(i)),ierr)
end do
end do
```

```
end if
call MPI_FINALIZE(ierr)
end
subroutine slave (a,n)
double precision a
integer n
a массивнинг локал қисмини қайта ишлаш
end
subroutine master(a,n)
double precision a
integer n
a массивни қайта ишлаш
end
MPI_TEST(REQUEST,FLAG,STATUS,IERR)
LOGICAL  FLAG
INTEGER REQUEST,IERR,STATUS(MPI_STATUS_SIZE)
```

REQUEST идентификатори билан боғланган MPI_ISEND ёки MPI_IRECV асинхрон амалнинг тугашини текшириш. FLAG параметрида қиймат қайтарилади. Агар амал тугаган бўлса, бу қиймат TRUE, акс холда FALSE га тенг бўлади. Агар қабул процедураси тугаган бўлса, у холда қабул қилинган хабар атрибутлари ва узунлигини STATUS параметри ёрдамида оддий тарзда аниқлаш мумкин. Процедура бажарилганидан кейин REQUEST параметрининг мос элемент MPI_REQUEST_NULL қийматида ўрнатилади.

```
MPI_TESTALL(COUNT,REQUESTS,FLAG,STATUSES,IERR)
LOGICAL FLAG
INTEGER COUNT,REQUESTS(*),STATUSES(MPI_STATUS_SIZE,*),IERR
```

REQUEST идентификатори билан боғланган асинхрон COUNT амалини тугашини текшириш. Агар кўрсатилган идентификатор билан

боғланган барча амаллар тугаган бўлса, FLAG параметрида процедура TRUE қийматини қайтаради. Бу холда хабар параметри STATUSES массивида кўрсатилади. Агар амаллардан бири тугамаган бўлса, у холда FALSE қиймати қайтарилади ва STATUSES массиви элементларининг аниқланганлиги кафолатланмайди. Процедура бажарилгандан кейин REQUEST параметрларининг мос элементлари MPI_REQUEST_NULL қийматида ўрнатилади.

MPI_TESTANY(COUNT,REQUESTS,INDEX,FLAG,STATUS,IERR)
LOGICAL FLAG
INTEGER
COUNT,REQUESTS(*),INDEX,FLAG,STATUS(MPI_STATUS_SIZE),IERR

REQUEST массивидаги идентификаторлар билан боғланган асинхрон амаллардан камида биттаси тугаганлигини текшириш. Агар асинхрон амаллардан ақалли биттаси тугаган бўлса, FLAG параметрида TRUE қиймати қайтарилади. Бунда INDEX ва REQUESTS массивидаги мос элементлар номери сақланади, STATUS да эса хабар параметрлари сақланади. Агар битта хам тугамаган бўлса, FALSE қиймати қайтарилади. Агар юқоридаги процедурасини чақириш вақтида бир неча кутилаётган амаллардан бир нечаси тугаган бўлса, у холда тасодифий равишда улардан бири танланади. Процедура бажарилгандан кейин REQUESTS параметрининг мос элементлари MPI_REQUEST_NULL қийматида кўрсатилади.

MPI_TESTSOME(INCOUNT,REQUESTS,OUTCOUNT,INDEXES,
STATUSES,IERR)
INTEGER
INCOUNT,REQUESTS(*),OUTCOUNT,INDEXES(*),IERR,STATUSES
(MPI_STATUS_SIZE,*)

Келтирилган процедурага MPI_WAITSOME процедураси ўхшаш бўлиб, бу процедура тез қайтарилади. Агар жорий процедурани чақириш

вақтигача тестланмаётган амалларнинг хеч бири тугамаган бўлса, у холда OUTCOUT қийматини 0 га тенг бўлади.

Навбатдаги мисолда жараёнлар ўртасида сатлари бўйича тақсимланган квадрат матрицани транспонирлашни бажарувчи блокировкаланмаган амалларни қўллаш намойиш этилган. Аввал хар бар жараён локал холда a массивнинг nl сатрини аниқлайди. Кейин MPI_ISEND ва MPI_IRECV блокировкаламайдиган амаллар ётдамида транспонирлаш учун зарур бўлган маълумотлар алмашунивлар белгиланади. Бошланган алмашинувлар фанида хар бир жараён ўзига тегишли A массивнинг локал қисмини транспонирлайди. Бундан кейин жараён MPI_WAITWAY процедурасини чақириш орқали бошқа ихтиёрий бир жараёндан хабар келишини кутади ва жорий жараёндан келган a массив қисмини транспонирлайди. Қайта ишлаш барча жараёнлардан хабар олиб бўлгунча давом этади. Охирида берилган a массив ва транспонирланган b массив чоп қилинади.

```
program example11
include 'mpif.h'
integer ierr,rank,size,N,nl,i,j
parametr (N=9)
double precision a(N,N),b(N,N)
call MPI_INIT(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD,size,ierr)
call MPI_COMM_RANK(MPI_COMM_RANK,rank,ierr)
n1=(N-1)/size+1
call work(a,b,n,n1,size,rank)
call MPI_FINALIZE(ierr)
end
```

Мулоқатнинг кечиктирилган сўровлари.

Жорий гурух процедуралари бир доирада узатиш қабул қилишдаги ортиқча харажатларни камайтиришга имкон яратади. Шунингдек жараёнлар ва тармоқ назорати ўртасида зарур маълумотлар алмашишдаги ортиқча уринишларни камайтиради. Кўпинча дастурда бир хил параметрли алмашинувларни кўп марта бажаришга тўғри келади. Бундай холда алмашиш амалини бир марта белгилаш ва кейин ундан маълумотларни ички структурасини белгилаш ва қўллаш учун ортиқча вақт йўқотмасдан хар бир яқинланишида кўп марта фойдаланиш мумкин. Бундан ташқари шу тарзда бир нечта узатиш қабул қилиш сўровларини битта буйруқ билан ишга тушириш учун бирлаштириш мумкин. Хабарни қабул қилиш усули уни жўнатиш усулига боғлиқ эмас. Юқорида келтирилган сўровлар ёки оддий усулда жўнатилган хабарлар оддий усулда хам ёки кечиктирилган сўровлар ёрдамида хам қабул қилиниши мумкин.

MPI_SEND_INIT(BUF,COUNT,DATATYPE,DEST,MSGTAG,COMM,
REQUEST,IERR)

<type>BUF(*)

INTEGER COUNT ,DATATYPE,DEST,MSGTAG,COMM,REQUEST,IERR

Хабар жўнатиш учун кечиктирилган сўровни расмийлаштириш. Бунда жўнатиш амали бошланмайди.

MPI_SEND ва MPI_ISEND, процедураларининг учта модификатциясига ўхшаш MPI_SEND_INIT процедурасининг учта қўшимча варианти мавжуд:

- MPI_BSEND_INIT буферлаб хабар жўнатиш учун кечиктирилган сўров расмийлаштириш;
- MPI_SSEND_INIT синхронлаб хабар жўнатиш учун кечиктирилган сўров расмийлаштириш;
- MPI_RSEND_INIT таёрлиги бўйича хабар жўнатиш учун кечиктирилган сўров расмийлаштириш.

MPI_RECV_INIT(BUF,COUNT,DATATYPE,SOURCE,MSGTAG,COMM, REQUEST,IERR)

<type> BUF(*)

INTEGER COUNT,DATATYPE,SOURCE,MSGTAG,COMM,REQUEST,IERR

REQUEST параметри қийматига мос маълумот алмашиш амалини бажариш учун кечиктирилган сўров яратиш. Амал блокировкаланмаган холда ишга тушади.

MPI_START(REQUEST,IERR)

INTEGER REQUEST,IERR

REQUEST массивининг дастлабки COUNT элементлари қийматига мос маълумотлар алмашиш амалларини бажариш учун COUNT та кечиктирилган сўровларни белгилаш. Амал блокировкаланмаган холда ишга тушади.

MPI_STARTALL(COUNT,REQUEST,IERR)

INTEGER COUNT,REQUEST,IERR

Блокировкаламайдиган амаллардан фарқли равишда кечиктирилган сўров ёрдамида ишга туширилган амал тугагандан кейин REQUEST параметри қиймати сақланиб қолади ва кейинчалик уни ишлатиш мумкин.

MPI_REQUEST_FREE(REQUEST,IERR)

INTEGER REQUEST,IERR

Жорий процедура REQUESTпараметри билан боғлиқ маълумотлар структурасини ўчиради. У бажарилгандан кейин REQUEST параметри MPI_REQUEST_NULL қийматида ўрнатилади. Агар ушбу сўров билан боғлиқ амал бажарилаётган бўлса, у холда унинг иши тугатилади.

Навбатдаги мисол халқа топологиясида қўшни жараёнлар ўртасида икки йўналишдаги амаллар учун кечиктирилган сўровлар ўрнатилади. Амалларнинг ўзи эса навбатдаги циклнинг хар бир яқинлашувида юкланади. Цикл тугаганидан кейин кечиктирилган сўровлар ўчирилади.

prev=rank-1

```
next=rank+1
if(rank.eq.0) prev=size-1
if(rank.eq.size-1) next=0
callMPI_RECV_INIT(rbuf(1),1,MPI_REAL,prev,5,
MPI_COMM_WORLD,reqs(1),ierr)
callMPI_RECV_INIT(rbuf(2),1,MPI_REAL,next,6,
MPI_COMM_WORLD,reqs(2),ierr)
callMPI_SEND_INIT(sbuf(1),1,MPI_REAL,prev,6,
MPI_COMM_WORLD,reqs(3),ierr)
callMPI_SEND_INIT(sbuf(2),1,MPI_REAL,next,5,
MPI_COMM_WORLD,reqs(4),ierr)
do i=...
  sbuf(1)=...
  sbuf(2)=...
call MPI_STARTALL(4,reqs,ierr)
  ...
call MPI_WAITALL(4,reqs,stats,ierr) ;
 ...
end do
call MPI_REQUEST_FREE(reqs(1),ierr)
call MPI_REQUEST_FREE(reqs(2),ierr)
call MPI_REQUEST_FREE(reqs(3),ierr)
call MPI_REQUEST_FREE(reqs(4),ierr)
```

2.3.Жараёнларнинг жамоа алоқаси

Жараёнларнинг жамоавий мулоқат алоқаларида коммуникатордаги барча жараёнлар қатнашади. Мос процедура ўзининг параметрлар тўплами билан барча жараёнлар томонидан чақирилади. Жамоавий мулоқат

процедурасидан қайтиш жараённинг жорий амалидаги иши тугаган пайтда юз беради. Блокировкаловчи процедурадаги каби қайтиш қабул ёки жўнатиш буферидан фойдаланиш мумкинлигини билдиради. MPI да асинхрон жамоа амаллар йўқ.

Жамоавий амаллар нуқта-нуқта тоифасидаги амаллар учун қўлланилган коммуникаторлардан фойдаланиш мумкин. MPI да жамоавий амаллар жараёнида чиқарилган хабарлар бошқа амалларни бажаришига таъсир қилмайди ва жараёнларнинг индивидуал мулоқатида пайдо бўлган хабарлар билан келишишмайди.

Умуман олганда жамоавий амаллар ёрдамида ишлаётган жараёнларнинг синхронлигига таянмаслик керак. Бирор жараёнлар иши хам тугаганлигини билдирмайди.

Жамоавий амалларда хабарлар идентификаторлари (тэглари) ишлатилмайди. Шу тарзда жамоавий амаллар дастур текстида жойлашишига мос холда қаътий тартибланган.

MPI_BARRIER(COMM,IERR)

INTEGER COMM, IERR

Процедура жараёнларни тўсиқли синхронлаш учун қўлланилади. Жараёнлар иши COMM коммуникаторидаги қолган жараёнлар жорий процедурани бажармагунича блокировкаланади. Коммуникатордаги охирги жараён жорий процедурани бажарганидан кейин барча жараёнлар блокировкадан бўшатилади ва бажарилишда давом этади. Жорий процедура жамоавий хисобланади. Коммуникаторнинг турли жараёнлари томонидан процедура мурожати дастурнинг турли қисмларидан бўлса хам барча жараёнлар MPI_BARRIER ни чақириши керак. Навбатдаги мисолда MPI_BARRIER процедурасининг вазифаси мулоқатлашишнинг кечиктирилган сўрови ёрдамида моделлаштирилади. Натижаларни ўртачасини аниқлаш учун NTIMES алмашиш амаллари бажарилади. Бунда

барча жараёнлар 0 номерли жараёнга хабар жўнатадилар ва шундан кейин 0 жараёндан барча жараёнлар дастурнинг жорий нуқтасигача етиб келганлигини билдирувчи жавоб сигнали олинади. Кечиктирилган сўровлардан фойдаланиш маълумотлар жўнатишни бир марта белгилашга ва кейин циклнинг хар бир яқинлашишида қўллашга имкон яратади.

```
program example13
include 'mpif.h'
integer ierr,rank,size,MAXPROC,NTIMES,i,it
parametr (MAXPROC =128,NTIMES= 1000)
integer ibuf(MAXPROC)
double precision time_start,time_finish
integer req(2*MAXPROC),statuses(MPI_STATUS_SIZE,MAXPROC)
call MPI_INIT(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD,size,ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)
if(rank .eq.0) then
do i=1,size-1
callMPI_RECV_INIT(ibuf(i),0,MPI_INTEGER,i,5,
MPI_COMM_WORLD,req(1),ierr)
callMPI_SEND_INIT(rank,0,MPI_INTEGER,i,6,
MPI_COMM_WORLD,req(size+1),ierr)
end do
time_start=MPI_WTIME(ierr)
do it =1,NTIMES
call MPI_STARTALL(size-1,req,ierr)
call MPI_WAITALL(size-1,req,statuses,ierr)
call MPI_STARTALL(size-1,req(size+1),ierr)
call MPI_WAITALL(size-1,req(size+1)statuses,ierr)
```

```
end do
else
callMPI_RECV_INIT(ibuf(1),0,MPI_INTEGER,0,6,
MPI_COMM_WORLD,req(1),ierr)
callMPI_SEND_INIT(rank,0,MPI_INTEGER,0,5,
MPI_COMM_WORLD,req(2),ierr)
time_start=MPI_WTIME(ierr)
do it =1,NTIMES
call MPI_START(req(2),ierr)
call MPI_WAIT(req(2),statuses,ierr)
call MPI_START(req(1),ierr)
call MPI_WAIT(req(1),statuses,ierr)
end do
end if
time_finish=MPI_WTIME(ierr)-time_start
print *,'rank=',rank,'all time = ', (time_finish)/NTIMES
time_start =MPI_WTIME(ierr)
do it=1,NTIMES
call MPI_BARRIER(MPI_COMM_WORLD,ierr)
enddo
time_finish=MPI_WTIME(ierr)-time_start
print *,'rank=',rank,'barrier time=', (time_finish)/NTIMES
call MPI_FINALIZE(ierr)
end
```

```
MPI_BCAST(BUF,COUNT,DATATYPE,ROOT,COMM,IERR)
<type>BUF(*)
INTEGER COUNT,DATATYPE,ROOT,COMM,IERR
```

ROOT жараёнидаги BUF массивидаги DATATYPE тоифали COUNT элементларини жорий COMM коммуникаторидаги барча жараёнларга жўнатиш. Процедурадан қайтгандан кейин ROOT жараёнининг BUF буферидаги маълумотлар COMM коммуникаторидаги хар бир жараённинг локал буферига нусхаланади. COUNT, DATATYPE, ROOT ва COMM параметрларининг қийматлари барча жараёнларда бир хил бўлиши керак.

Масалан, 2 жараёндан дастурнинг қолган жараёнларига BUF массивини ва 100 бутун сонли элементларни жўнатиш учун барча жараёнларда қуйидаги чақирув мавжуд бўлиши керак:

call MPI_BCAST(BUF,100 ,MPI_INTEGER,2,MPI_COMM_WORLD,ierr)

MPI_GATHER(SBUF,SCOUNT,STYPE,RBUF,RCOUNT,RTYPE,ROOT,
COMM,IERR)

<type>SBUF(*),RBUF(*)

INTEGER SCOUNT,STYPE,RCOUNT,RTYPE,ROOT,COMM,IERR

STYPE тоифали SCOUNT маълумотлар элементларини COMM коммуникаторидаги барча жараёнларнинг SBUF массивларидан олиб ROOT жараёнининг RBUF массивида тўплаш. ROOT жараёни ва қолган жараёнлар ўзининг SBUF буферидаги маълумотларни жўнатган жараёнлар номерларининг тартибида RBUF буферида жойлаштирилади.

ROOT жараёни учун барча параметрлар қийматлари ахамиятга эга, қолган жараёнлар учун SBUF, SCOUNT, STYPE, ROOT ва COMM параметрларининг қийматлари ахамиятга эга. ROOT ва COMM параметрининг қийматлари барча жараёнларда бир хил бўлиши керак. ROOT жараёнининг RCOUNT параметри барча жараёнлардан эмас, балки хар бир жараёнлардан келадиган RTYPE тоифали элементлар миқдорини билдиради. Масалан, 2 жараён rbuf массивида хар бир жараённинг buf массивидаги 10 та бутун сонли элементларни тўплаши учун барча жараёнларда қуйидаги чақирув мавжуд бўлиши керак:

callMPI_GATHER(buf,10,MPI_INTEGER,rbuf,10,MPI_INTEGER,2,
MPI_COMM_WORLD,ierr)
MPI_GATHERV(SBUF,SCOUNT,STYPE,RBUF,RCOUNTS,DISPLS,RTYPE,
ROOT,COMM,IERR)
<type> SBUF(*),RBUF(*)
INTEGER
SCOUNT,STYPE,RCOUNTS(*),DISPLS(*),RTYPE,ROOT,COMM,IERR

SBUF массивидаги турли миқдордаги маълумотларни тўплаш. Натижавий RBUF буферида маълумотларни жойлашиш тартиби DISPLS массиви белгилайди.

RCOUNTS бутун сонли массив бўлиб, хар бир жараёндан жўнатиладиган элементлар миқдорини билдиради (индекс жўнатувчи жараён рангига массив ўлчами COMM коммуникаторидаги жараёнлар сони тенг).
DISPLS бутун сонли массив бўлиб, RBUF массив бошига нисбатан силжишини сақлайди (индекс жўнатувчи жараён рангига тенг, массив ўлчами эса COMM коммуникаторидаги жараёнлар сонига тенг).
J-1 жараён жўнатган маълумотлар ROOT жараёни RBUF буферининг J-блокида жойлашади.
MPI_SCATTER(SBUF,SCOUNT,STYPE,RBUF,RCOUNT,RTYPE,ROOT,
COMM,IERR)
<type>SBUF(*),RBUF(*)
INTEGER SCOUNT,STYPE,RCOUNT,RTYPE,ROOT,COMM,IERR

MPI_SCATTER процедураси бажарадиган амалига кўра MPI_GATHER га тескари хисобланади. У ROOT жараёнидаги SBUF массивидан STYPE тоифали SCOUNT тадан иборат элементларни ROOT жараёни ва COMM коммуникаторидаги жараёнларнинг RBUF массивига жўнатади. SBUF буфери жараёнлар сонига тенг бўлакларга бўлинади ва уларнинг хар бирига STYPE тоифали SCOUNT элемент мавжуд. Унда i-қисм (i-1)- жараёнга

жўнатилади.

ROOT жараён учун барча параметрлар ахамиятли бўлади. Қолган жараёнлар учун фақат RBUF,RCOUNT,RTYPE,SOURCE ва COMM параметрлари қийматлари зарур бўлади. SOURCE ва COMM қийматлари барча жараёнлар учун бир хил бўлиши керак.

Навбатдаги мисолда 0-жараёни sbuf массивини аниқлайди ва бундан кейин уни алохида устунларини дастурда ишга тушган алохида жараёнларга жўнатади. Хар бир жараёндаги rbuf массивида жойлашади.

```
real sbuf(size,size),rbuf(size)
if(rank .eq. 0) then
    do  1 i=1,size
      do 1 j=1,size
 1    sbuf(i,j)=...
end if
if(numtasks .eq. size) then
callMPI_SCATTER(sbuf,size,MPI_REAL,rbuf,size,
MPI_REAL,0,MPI_COMM_WORLD,ierr)
end if
```

```
MPI_SCATTER(SBUF,SCOUNT,DISPLS,STYPE,RCOUNT,RTYPE,ROOT,
COMM,IERR)
<type>SBUF(*),RBUF(*)
INTEGER
SCOUNTS(*),DISPLS(*),STYPE,RCOUNT,RTYPE,ROOT,COMM,IERR
```

SBUF массивида турли миқдордаги маълумотларни жўнатиш. Жўнатиладиган маълумотлар бўлаги, боши DISPLS массивида берилади. SCOUNTS бутун сонли массив бўлиб, хар бир жараёнга жўнатиладиган элементлар миқдорини сақлайди (индекс манзил жараён рангига, узунлиги COMM коммуникаторидаги жараёнлар миқдорига тенг).

ROOT жараёнининг J-1 жараёнга жўнатадиган маълумотлар SBUF буферининг J инчи блокидан жойлашган бўлиб, DISPLS(J) силжишидан бошланади. SBUF буфер билан STYPE бошланади.

MPI_ALLGATHER(SBUF,SCOUNT,STYPE,RBUF,RCOUNT,RTYPE,
COMM,IERR)

<type>SBUF(*),RBUF(*)

INTEGER SCOUNT,STYPE,RCOUNT,RTYPE,COMM,IERR

COMM коммуникаторининг барча жараёнларида SBUF массивларидаги маълумотларни хар бир жараённинг RBUF буферида тўплаш. Маълумотлар жараёнлар номерининг ўсиш тартибида жойлашади. J-1 инчи жараён жўнатган маълумотлар блоки қабул қилувчи жараённинг RBUF буферидаги J инчи блокида жойлашади. Бу амални MPI_GATHER каби қараш мумкин, унинг бажарилиш натижаси COMM коммуникаторининг барча жараёнларида хосил бўлади.

MPI_ALLGATHERV(SBUF,SCOUNT,STYPE,RBUF,RCOUNTS,DISPLS,
RTYPE,COMM,IERR)

<type>SBUF(*),RBUF(*)

INTEGER SCOUNT,STYPE,RCOUNTS(*),DISPLS(*),RTYPE,COMM,IERR

COMM коммуникаторининг барча жараёнларида SBUF массивларидан турли миқдорда маълумотларни тўплаш. RBUF массивида маълумотларни жойлашиш тартиби DISPLS массивида берилади.

MPI_ALLTOALL(SBUF,SCOUNT,STYPE,RBUF,RCOUNT,RTYPE,
COMM,IERR)

<type>SBUF(*),RBUF(*)

INTEGER SCOUNT,STYPE,RCOUNT,RTYPE,COMM,IERR

COMM коммуникаторининг хар бир жараёни томонидан келган барча жараёнларга турли улушли маълумотларни жўнатиш. (I-1) инчи жараённинг SBUF буферидаги J инчи маълумотлар блоки (J-1) инчи жараённи RBUF

буферининг I инчи маълумотлар блокига жойлашади.

MPI_ALLTOALL(SBUF,SCOUNT,SDISPLS,STYPE,RBUF,RCOUNTS,
RDISPLS,RTYPE,COMM,IERR)

<type>SBUF(*),RBUF(*)

INTEGER RCOUNTS(*),SDISPLS(*),STYPE.SCOUNTS(*),RDISPLS(*),
RTYPE,COMM,IERR

COMM коммуникаторининг барча жараёнларидан жорий коммуникаторнинг барча жараёнларига турли миқдордаги маълумотларни жўнатиш. Жўнатувчи жараённинг SBUF буферида маълумотлар жойлашиш ўрни SDISPLS массивида берилади. Қабул қилувчи жараённинг RBUF буферида маълумотларни жойлашиш ўрни RDISPLS массивида берилади.

MPI_REDUCE(SBUF,RBUF,COUNT,DATATYPE,OP,ROOT,COMM,IERR)

<type> SBUF(*),RBUF(*)

INTEGER COUNT,DATATYPE,OP,ROOT,COMM,IERR

SBUF массивларининг мос элементлари устида COUNT та мустақил OP глабол амалларни бажариш. COMM коммуникаторидаги барча жараёнларини SBUF массивининг I инчи элементи устида бажарилган OP амални бажарилиш натижаси ROOT жараённинг RBUF массивидаги I инчи элементида хосил бўлади(жойланади).

MPI да бир қатор глабол амаллар аниқланган бўлиб, улар қуйидаги константалар билан берилади.

- MPI_MAX,MPI_MIN- максимум ва минимум қийматини аниқлаш;
- MPI_MINLOG,MPI_MAXLOG- максимум ва минимум қийматларни ва уларни жойлашиш ўринини аниқлаш;
- MPI_SUM,MPI_PROD-глобал йиғинди ва глобал кўпайтмани аниқлаш;
- MPI_LAND,MPI_LOR,MPI_LXOR-мантиқий "ва", "ёки" ;
- MPI_BAND,MPI_BOR,MPI_BXOR-битли "ва", "ёки".

Бундан ташқари дастурчи глобал амалларни бажариш учун

MPI_OP_GREATE процедураси ёрдамида ўз функциясини бериш мумкин.

Навбвтдаги мисолда глобал суммалаш амали нуқта-нуқта усулида маълумотларни жўнатишни қўлланган иккилантириш схемаси ёрдамида моделлаштирилади. Бундай моделлаш эффективлиги MPI_REDUCE жамоавий амални қўллаб солиштиради:

```
program example14
include 'mpif.h'
integer ierr,rank,i,size,nproc
parametr (n=1 000 000)
double precision a(n),b(n),c(n)
integer status (MPI_STATUS_SIZE)
call MPI_INIT(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD,size,ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)
nproc=size
do i=1,n
a(i)=1.do/size
end do
call MPI_BARRIER(MPI_COMM_WORLD,ierr)
time_start=MPI_WTIME(ierr)
do i=1,n
c(i)=a(i)
end do
do while (nproc .gt.1)
if(rank .lt. nproc/2) then
callMPI_RECV(b,n,MPI_DOUBLE_PRECISION,nproc-rank-1,1,
MPI_COMM_WORLD,status,ierr)
do i=1,n
```

```
c(i)=c(i)+b(i)
end do
else if(rank .lt. nproc) then
callMPI_SEND(c,n,MPI_DOUBLE_PRECISION,nproc-rank-1,1,
MPI_COMM_WORLD,ierr)
end if
nproc=nproc/2
end do
do i=1,n
b(i)=c(i)
end do
time_finish=MPI_WTIME(ierr)-time_start
if(rank .eq.0) print *,'model b(1)=',b(1)
print *, 'rank=',rank,'model time=', time_finish
do i=1,n
a(i)=1.do/size
end do
call MPI_BARRIER(MPI_COMM_WORLD,ierr)
time_start=MPI_WTIME(ierr)
callMPI_REDUCE(a,b,n,MPI_DOUBLE_PRECISION,MPI_SUM,0,
MPI_COMM_WORLD,ierr)
time_finish=MPI_WTIME(ierr)-time_start
if(rank .eq. 0) print *, 'reduce b(1)=', b(1)
print *, 'rank =', rank, 'reduce time =', time_finish
call MPI_FINALIZE (ierr)
end
```

MPI_ALLREDUCE(SBUF,RBUF,COUNT,DATATYPE,OP,COMM,IERR)

<type>SBUF(*),RBUF(*)

INTEGER COUNT,DATATYPE,OP,COMM,IERR

SBUF массивининг мос элементлари устида COUNT та мустақил глобал OP амалларини бажариш MPI_REDUCE процедурасидан фарқли томони шундаки, натижа хар бир жараённинг RBUF массивида хосил бўлади. Навбатдаги мисолда хар бир жараён локал массивнинг сатр йиғиндиларини хисоблайди, кейин барча жараёнларда хосил бўлган йиғиндилар MPI_ALLREDUCE процедураси ёрдамида тўпланади ва натижа дастурдаги барча жараёнларнинг r массивига ёзилади.

```
do i=1,n
  s(i)=0.0
end do
do i=1,n
   do j=1,n
s(i)=s(i)+a(i,j)
end do
end do
call MPI_ALLREDUCE(s,r,n,MPI_REAL,MPI_SUM,MPI_COMM_WORLD,
IERR)
```

MPI_REDUCE_SCATTER(SBUF,RBUF,RCOUNTS,DATATYPE,OP,
COMM,IERR)

<type>SBUF(*),RBUF(*)

INTEGER RCOUNTS(*),DATATYPE,OP,COMM,IERR

SBUF массивларнинг мос элементлари устида RCOUNTS(I) мустақил глобал OP амалларни бажариш. Вазифасига кўра бу амал аввал глобал амалларни бажарилиши ва хосил бўлган натижалар жараёнларга жўнатилишига тенг кучли. I инчи жараён натижасининг RCOUNTS(I+1)

элементдан иборат (I+1) улушини олади ва RBUF массивига жойлайди. RCOUNTS массиви COMM коммуникаторидаги барча жараёнларда бир хил бўлиши керак.

MPI_SCAN(SBUF,RBUF,COUNT,DATATYPE,OP,COMM,IERR)

<type>SBUF(*),RBUF(*)

INTEGER COUNT,DATATYPE,OP,COMM,IERR

SBUF массивларининг мос элементлари устида COUNT та мустақил глобал OP амалларни бажаради. I инчи жараён 0 дан I гача бўлган номерли жараёнларнинг SBUF массивларининг мос элементлари устида COUNT та глобал амални бажаради ва олинган натижани RBUF массивида жойлайди. Глобал амалнинг тўлиқ натижаси охирги жараённинг RBUF массивида хосил бўлади.

MPI_OP_GREATE(FUNC,COMMUTE,OP,IERR)

EXTERNAL FUNC

LOGICAL COMMUTE

INTEGER OP,IERR

FUNC функцияси ёрдамида ишга тушадиган фойдаланувчининг OP глобал амалини яратиш. Яратиладиган амал ассоциатив бўлиши керак. Агар COMMUTE парвметри қиймати TRUE бўлса амал коммутатив хам бўлиши керак. Агар COMMUTE параметри қиймати FALSE бўлса, у холда глобал амални бажарилиш тартиби 0 инчи номерли жараёндан бошлаб жараёнлар номерларини ўсиб бориш тартибида қаттий белгиланади.

FUNCTION FUNC(INVEC(*),INOUTVEC(*),LEN,TYPE

<type>INVEC(LEN),INOUTVEC(LEN)

INTEGER LEN,TYPE

Шу тарзда глобал амални яратишда фойдаланувчи функциясининг интерфейси яратилади.

Амалнинг биринчи аргументи INVEC параметридан олинади, иккинчи

аргументи INOUTVEC параметридан олинади. Натижа эса INOUTVEC параметрида қайтарилади. LEN параметри кирувчи ва чиқувчи массивлар элементларининг миқдорини, TYPE параметри эса кирувчи ва чиқувчи маълумотлар тоифасини билдиради. Фойдаланувчи функциясида MPI процедуралари ёрдамида хеч қандай маълумотлар алмашиши амалга оширилмаслиги лозим.

MPI_OP_FREE(OP,IERR)

INTEGER OP,IERR

Фойдаланувчи глобал амалини йўқ қилиш. Процедура бажарилгандан кейин OP ўзгарувчига MPI_OP_NULL қиймати берилади.

Навбатдаги мисолда глобал амал сифатида қўллаш учун фойдаланувчи функциясини берилиши намойиш этилган. Унда s mod 5 функцияси берилган бўлиб, у модул бўйича элементлар йиғиндисини хисоблайди. Жорий функция MPI_OP_CREAT процедура чақирувида глобал OP амали сифатида эълон қилинади, кейин MPI_REDUCE процедурасида қўлланилади. Шундан кейин MPI_OP_FREE процедурасини чақириш орқали ўчирилади.

```
program example15
include 'mpif.h'
integer ierr,rank,i,n
parametr (n=1 000)
integer a(n),b(n)
integer op
external smod5
call MPI_INIT(ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)
do i=1,n
a(i)=i+rank
end do
```

```
print *, 'process',rank ,'a(1)=',a(1)
call MPI_OP_CREATE(smod5, .TRUE.,op,ierr)
call MPI_REDUCE(a,b,n,MPI_INTEGER,op,0, MPI_COMM_WORLD,ierr)
call MPI_OP_FREE(op,ierr)
if(rank .eq.0) print *,'b(1)=',b(1)
call MPI_FINALIZE(ierr)
end

integer function smod5(in,inout,1,type)
integer 1,type
integer in(1),inout(1),i
do i=1,1
inout(i)=mod(in(i)+inout(i),5)
end do
return
end
```

2.4.Гурухлар ва коммуникаторлар.

MPI да жараёнлар ва коммуникаторлар гурухлари устида амллар бажаришнинг кенг имкониятлари мавжуд. Бу биринчидан бирор жараёнлар гурухига ўзининг қисм масаласи устида мустақил ишлаш имконини бериш учун зарур. Иккинчидан, агар алгоритм хусусиятига кўра жараёнларнинг фақат бир қисмигина маълумотлар алмашиш керак бўлса, уларни маълумотлар алмашиш учун алохида коммуникатор яратиш қулай хисобланади. Учинчидан, қисм дастурлар кутубхоналарини яратишда кутубхона модуллари ичидаги маълумотлар алмашиш асосий дастурдаги маълумотлар алмашиш билан кесишмаслиги кафолатланиши керак. Бу масалани тўла хажмда мустақил коммуникаторларни яратиш орқали ечиш

мумкин.

Гурух – бу жараёнларнинг тартибланган тўплами. Гурухдаги хар бир жараёнга бутун сон-ранг ёки номер мос қўйилади. MPI_GROUP_EMPTY битта хам жараёнга эга бўлмаган бўш гурухдир. MPI_GROUPE_NULL хато гурухлар учун қўлланиладиган қиймат.

Янги гурухлар мавжуд гурухлар асосида хам ва коммуникаторлар асосида хам яратилиши мумкин. Бироқ маълумотлар алмашиш амалларида фақат коммуникаторлар қатнашади. Жараёнларнинг барча гурухлари яратиладиган асосий гурух MPI_COMM_WORLD коммуникатори билан боғланган. Унга дастурнинг барча жараёнлари тегишли хисобланади. Жараёнлар гурухлари устидаги амаллар локал хисбланади. Унга процедурани чақирувчи жараёнлар киритилади. Процедурани бажаришда жараёнлар ўртасида маълумотлар алмашиш талаб этилмайди. Ихтиёрий жараёнга ихтиёрий гурух устида, шунингдек жорий жараён тегишли бўлмаган гурух устида хам амаллар бажариш мумкин. Гурухлар устида амаллар бажаришда MPI_GROUP_EMPTY бўш гурухи хам пайдо бўлиши мумкин.

MPI_COMM_GROUP(COMM,GROUP,IERR)

INTEGER COMM, GROUP,IERR

COMM коммуникаторига мос GROUP гурухи хосил қилиш. Си тилида GROUP параметри MPI_GROUP тоифага тегишли бўлади. Қолаверса, дастлаб ягона MPI_COMM_WORLD коммуникатори мавжуд бўлади. Аввал унга мос жараён гурухини аниқлаш зарур. Бу қуйидагича аниқланади:

call MPI_COMM_GROUP(MPI_COMM_WORLD,group,ierr)

MPI_GROUP_INCL(GROUP,N,RANKS,NEWGROUP,IERR)

INTEGER GROUP,N,RANKS(*),NEWGROUP,IERR

Аввал GROUP гурухидаги N жараёндан RANKS(1),...,RANKS(N) рангли янги NEWGROUP гурухини яратиш. Бунда эски гурухдаги RANKS(I) рангига янги гурухдаги I-1 инчи ранг тўғри келади. N=0 да бўш бўлган

MPI_GROUP_EMPTY гурухи яратилади. Бу процедура гурухда янги тартибдаги жараёнларни яратиш учун қўлланиши мумкин.

MPI_GROUP_EXCL(GROUP,N,RANKS,NEWGROUP,IERR)

INTEGER GROUP,N,RANKS(*),NEWGROUP,IERR

GROUP гурихидаги RANK(1),...,RANK(N) рангли жараёнлардан NEWGROUP гурухини яратиш. Бунда янги гурухдаги жараёнлар тартиби жараёнларнинг эски гурухидаги тартибига мос келади. N=0 да эски гурухга ўхшаш гурух яратилади.

Навбатдаги мисолда group гурухи жараёнлари асосида кесишмайдиган иккита group1 ва group2 жараёнлар гурухи яратилади. Яратилган хар бир гурухга аввалги гурухдаги жараёнларнинг тахминан ярми киради (жараёнлар сони тоқ бўлганда group2 гурухига “битта” кўп жараён киради). Янги яратилган гурухлардаги жараёнларни номерлаш тартиби сақланади.

```
size1=size/2
do  i=1,size1
  ranks(i)=i-1
enddo
call MPI_GROUP_INCL(group,size1,ranks,group1,ierr)
call MPI_GROUP_EXCL(group,size1,ranks,group2,ierr)
```

Навбатдаги учта процедура худди тўпламлар каби жараёнлар гурухи устида бажариладиган амаллар аниқлайди. Жараёнларни номерлаш ўзига хослиги туфайли бирлашмаси хам, кесишмаси хам коммутатив эмас, бироқ ассоциатив.

MPI_GROUP_INTERSECTION(GROUP1,GROUP2,NEWGROUP,IERR)

INTEGER GROUP1,GROUP2,NEWGROUP,IERR

GROUP1 ва GROUP2 гурухларнинг кесишмасидан NEWGROUP гурухини яратиш. Янги гурух таркибига GROUP1 гурухининг GROUP2 гурухига хам тегишли бўлган жараёнлари биринчи гурухдаги тартиби билан

киритилади.

MPI_GROUP_UNION(GROUP1,GROUP2,NEWGROUP,IERR)

INTEGER GROUP1,GROUP2,IERR

GROUP1 ва GROUP2 гурухларнинг бирлашмасидан NEWGROUP гурухини яратиш. Хосил бўлган гурух таркибига GROUP1 гурухидаги жараёнлар ўз тартибида ва GROUP2 гурухидаги GROUP1 гурухига хам тегишли бўлмаган гурухлари ўз тартибида киритилади.

MPI_GROUP_DIFFERENCE(GROUP1,GROUP2,NEWGROUP,IERR)

INTEGER GROUP1,GROUP2,NEWGROUP,IERR

GROUP1 ва GROUP2 гурухларининг фарқидан NEWGROUP гурухини яратиш. Янги гурухга GROUP1 гурухидаги GROUP2 гурухига кирмаган ва биринчи гурухдагидек тартибланган жараёнлар киритилади.

Масалан, gr1 гурухига 0,1,2,3,4,5 жараёнлари, gr2 гурухига 0,2,3 жараёнлари (жараёнлар номерлари MPI_COMM_WORLD коммуникаторидаги гурухга мос берилган). У холда қуйидаги

call MPI_GROUP_INTERSECTION(gr1,gr2,newgr1,ierr)

call MPI_GROUP_UNION(gr1,gr2,newgr2,ierr)

call MPI_GROUP_DIFFERENCE(gr1,gr2,newgr3,ierr)

newgr1 гурухига 0,2 жараёни киради;

newgr2 гурухига 0,1,2,4,5,3 жараёнлари киради;

newgr3 гурухига 1,4,5 жараёнлари киради.

Янги гурухларда жараёнларни номерлаш тартиби уларни санаш тартибига мос келади.

MPI_GROUP_SIZE(GROUP,SIZE,IERR)

INTEGER GROUP,SIZE,IERR

GROUP гурухидаги жараёнларнинг SIZE миқдорини аниқлаш.

MPI_GROUP_RANK(GROUP,RANK,IERR)

INTEGER GROUP,RANK,IERR

GROUP гурухидаги жараённинг RANK номерини аниқлаш. Агар процедурани чақирувчи жараён GROUP гурухига тегишли бўлмаса, у холда MPI_UNDEFINED қиймати қайтарилади.

MPI_GROUP_TRANSLATE_RANKS(GROUP1,N,RANKS1,GROUP2,
RANKS2,IERR)

INTEGER GROUP1,N,RANKS1(*),GROUP2,RANKS2(*),IERR

RANK2 массивида GROUP2 гурухидаги жараёнлар ранглари қайтарилади. N параметри ранги аниқланиши керак бўлган жараёнлар миқдорини билдиради.

MPI_GROUP_COMPARE(GROUP1,GROUP2,RESULT,IERR)

INTEGER GROUP1,GROUP2,RESULT,IERR

GROUP1 ва GROUP2 гурухларини солиштириш. Агар GROUP1 ва GROUP2 гурухлари тўлиқ мос келса, у холда RESULT параметрида MPI_IDENT қиймати қайтарилади. Агар гурухлар жараёнлари ранглари билан фарқ қилса, у холда MPI_SIMILAR қиймати қайтарилади. Акс холда MPI_UNEQUAL қиймати қайтарилади.

MPI_GROUP_FREE(GROUP,IERR)

INTEGER GROUP,IERR

GROUP гурухини ўчириш. Процедура бажарилгандан кейин GROUP ўзгарувчисига MPI_GROUP_NULL қиймати берилади. Агар ушбу процедура чақирилиш пайтида жорий гурух билан бир амал бажарилаётган бўлса, у холда бу амал тугатилади.

Навбатдаги мисолда дастурнинг барча жараёнлари тахминан ўзаро тенг бўлган group1 ва group2 гурухларига тақсимланади. Жараёнлар миқдори тоқ бўлганда group2 да битта жараён кўп бўлади. Бу холда жорий гурухда бу жараён group1 гурухидаги хеч бир жараён билан маълумот алмашмаслиги керак. Хар бир жараён MPI_GROUP_TRANSLATE_RANKS процедурасини чақириш орқали бошқа гурухдаги бир хил номерли жараённи аниқлайди ва у

билан MPI_SENDRECV процедурасини чақириб MPI_COMM_WORLD коммуникатори орқали хабар алмашади. Дасдур охирида кейинчалик керак бўлмаган гурухлар MPI_GROUP_FREE процедурасини чақириш орқали ўчиради.

```
program example16
include 'mpif.h'
integer ierr,rank,i,size,size1
integer a(4),b(4)
integer status(MPI_STATUS_SIZE)
integer group,group1,group2
integer ranks(128),rank1.rank2,rank3
call MPI_INIT(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD,size,ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)
call MPI_COMM_GROUP(MPI_COMM_WORLD,group,ierr)
size1=size/2
do i=1,size1
  ranks(i)=i-1
enddo
call MPI_GROUP_INCL(group,size1,ranks,group1,ierr)
call MPI_GROUP_EXCL(group,size1,ranks,group2,ierr)
call MPI_GROUP_RANK(group1,rank1,ierr)
call MPI_GROUP_RANK(group2,rank2,ierr)
if(rank1 .eq. MPI_UNDEFINED) then
if(rank2 .lt.  size1) then
call MPI_GROUP_TRANSLATE_RANKS(group1,1.rank2,group,rank3,ierr)
else
rank3=MPI_UNDEFINED
```

```
end if
else
call MPI_GROUP_TRANSLATE_RANKS(group2,1.rank1,group,rank3,ierr)
end if
a(1)=rank
a(2)=rank1
a(3)=rank2
a(4)=rank3
if(rank3.ne.MPI_UNDEFINED) then
callMPI_SENDRECV(a,4, MPI_INTEGER,rank3,1,b,4, MPI_INTEGER,rank3,1,
MPI_COMM_WORLD,status,ierr)
 end if
call MPI_GROUP_FREE(group,ierr)
call MPI_GROUP_FREE(group1,ierr)
call MPI_GROUP_FREE(group2,ierr)
print *,'process',rank,'a=',a,'b=',b
call MPI_FINALIZE(ierr)
end
```

Коммуникаторлар билан амал бажариш.

Коммуникатор бирор гурух жараёнлари маълумот алмашишда контекстни таъминлаб беради. Контекст маълумотларни мустақил алмашиш имконини таъминлаб беради. Жараёнларнинг хар бир гурухига бир неча коммуникаторлар мос келади, лекин хар бир коммуникатор ихтиёрий вақт пайтида битта гурухга тегишли бўлади.

Навбатдаги коммуникаторлар MPI_INIT процедураси чақирилиши билан дарров вужудга келади:

- MPI_COMM_WORLD дастурдаги барча жараёнларни бирлаштирувчи

коммуникатор;

- MPI_COMM_NULL хато коммуникатор учун қўлланувчи қиймат;
- MPI_COMM_SELF фақат чақирувчи жараённи ўзига олувчи коммуникатор.

Коммуникатор яратиш жамоавий амал хисобланади ва процессорлар ўртасида маълумот алмашишни талаб этади. Шунинг учун бундай процедуралар бирор мавжуд коммуникаторнинг барча жараёнлари томонидан чақирилиши мумкин.

```
MPI_COMM_DUP(COMM,NEWCOMM,IERR)
INTEGER COMM,NEWCOMM,IERR
```

COMM коммуникаторидаги жараёнлари гурухларига ва атрибутига эга бўлган NEWCOMM коммуникаторини яратиш.

```
MPI_COMM_CREATE(COMM,GROUP,NEWCOMM,IERR)
INTEGER COMM,GROUP,NEWCOMM,IERR
```

COMM коммуникаторидан GROUP жараёнлар гурухи учун NEWCOMM янги коммуникаторини яратиш. GROUP гурухи COMM коммуникатори билан боғланган гурухнинг бир қисми бўлиши керак. Бу чақирув COMM коммуникаторининг барча жараёнларида учраши керак. GROUP гурухига тегишли бўлмаган жараёнларда MPI_COMM_NULL қиймати қайтарилади.

Навбатдаги мисолда иккита янги гурух яратилади. Улардан бирига жараёнларнинг биринчи ярим, иккинчисига жараёнларнинг иккинчи ярим тегишли бўлади. Жараёнлар сони тоқ бўлганда иккинчи гурухда жараёнлар сони битта кўп бўлади. Хар бир янги гурух учун мос new_comm коммуникатори яратилади ва MPI_ALLREDUCE амали турли гурухларга тегишли бўлган жараёнлар учун алохида бажарилади.

```
call MPI_COMM_GROUP(MPI_COMM_WORLD,group,ierr)
do i=1,size/2
```

```
  ranks(i)=i-1
end do
if(rank .1t. size/2) then
call MPI_GROUP_INCL(group,size/2,ranks,new_group,ierr)
else
call MPI_GROUP_EXCL(group,size/2,ranks,new_group,ierr)
end if
call MPI_COMM_CREATE(MPI_COMM_WORLD,new_group,new_comm,ierr)
call MPI_ALLREDUCE(sbuf,rbuf,1,MPI_INTEGER,MPI_SUM,new_comm,ierr)
call MPI_GROUP_RANK(new_group,new_rank,ierr)
print *,'rank=',rank,'newrank=',new_rank,'rbuf=',rbuf
MPI_COMM_SPLIT(COMM,COLOR,KEY,NEWCOMM,IERR)
INTEGER COMM,COLOR,KEY,NEWCOMM,IERR
```

COMM коммуникатори COLOR параметри қиймати бўйича бир неча янги коммуникаторларга бўлинади. Битта коммуникаторга COLOR қийматлари бир хил бўлган жараёнлар киради. KEY параметри қиймати катта бўлган жараёнлар янги гурухда катта рангга эга бўлади. KEY параметри қийматлари тенг бўлганда жараёнларни номерлаш система томонидан танланади.

Янги коммуникаторга кирмаслиги керак бўлган жараёнларда COLOR параметри қиймати сифатида MPI_UNDEFINED константаси кўрсатилади. Уларга NEWCOMM параметрида MPI_COMM_NULL қиймати қайтирилади.

Навбатдаги мисолда MPI_COMM_WORLD коммуникатори учта бўлакка бўлинади. Биринчисига 0,3,6 ва хаказолар. Номерли жараёнлар, иккинчисига 1,4,7 хаказо ва учинчисига 2,5,8 ва хаказо жараёнлар киради. KEY параметри сифатида rank ўзгарувчисини кўрсатилиши яратиладиган гурухларда жараёнларни номерлаш тартиби бошланғич гурухдаги номерлаш тартибига мос келишини кафолатлайди.

```
callMPI_COMM_SPLIT(MPI_COMM_WORLD,mod(rank,3),rank,
new_comm,ierr)
MPI_COMM_FREE(COMM,IERR)
INTEGER COMM,IERR
```

COMM коммуникаторини ўчириш. Бу процедура бажарилганидан кейин COMM ўзгарувчига MPI_COMM_NULL қиймати берилади. Агар бу процедурани чақириш вақтида COMM коммуникатори билан процедура амал бажараётган бўлса, бу амал тугатилади.

Навбатдаги мисолда битта янги comm_revs коммуникатори яратилади ва унга дастурдаги барча жараёнлар тескари тартибда номерлаб киритилади. Бу коммуникаторга зарурат бўлмай қолганда, у MPI_COMM_FREE процедураси чақирилиб ўчирилади. MPI_COMM_SPLIT процедурасини жараёнларни қайта номерлаш учун ишлатиш мумкин.

```
program exampl17
include 'mpif.h'
integer ierr,rank,size
integer comm_revs,rank1
call MPI_INIT(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD,size,ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)
callMPI_COMM_SPLINT(MPI_COMM_WORLD,1,size-rank,
comm_revs,ierr)
call MPI_COMM_RANK(comm_revs,rank1,ierr)
print*,'rank=',rank,'rank1=',rank1
call MPI_COMM_FREE(comm_revs,ierr)
call MPI_FINALIZE(ierr)
end
```

2.5.Турли тоифали маълумотларни узатиш

MPI да хабар деганда хотиранинг кетма-кет ячейкаларда жойлашган бир хил тоифали маълумотлар массиви тушинилади. Кўпинча дастурларда маълумотларнинг, нисбатан мураккаб хар хил тоифали элементлардан иборат тузилмасини ёки хотиранинг кетма-кет ячейкаларида жойлашмаган маълумотларни жўнатиш зарур бўлади. Бундай холларда маълумотларни кетма-кет жойлашган бир хил тоифали элементлардан иборат катта бўлмаган қисмлар шаклида ёки маълумотларни жўнатишдан олдин вақтинчалик бирор буферга нусхалаш керак. Иккила усул хам етарлича ноқулай ва вақт тезкор хотирани қўшимча сарфлашни талаб этади.

MPI да хар хил тоифали маълумотларни жўнатиш учун иккита махсус усул мавжуд:

- Маълумотларнинг хосилавий тоифаси;
- Маълумотларнинг ихчам жойлаш ёки ўраш;

Маълумотларнинг хосилавий тоифаси.

Маълумотларнинг хосилавий тоифаси дастурни бажариш вақтида конструктор процедуралар ёрдамида мажуд тоифалар асосида яратилади.

Маълумотлар тоифасини яратиш иккита босқичдан иборат:

1. Тоифасини конструкциялаш;
2. Тоифасини рўйхатдан ўтказиш;

Хосилавий тоифа рўйхатдан ўтгандан кейин уни жўнатиш ва жамоавий амалларда мавжуд тоифалар билан бир қаторда ишлаш мумкин. Хосилавий тоифали маълумотлар билан боғлиқ иш тугагандан кейин уни йўқотиш керак. Бунда хосилавий тоифа сақланиб қолади ва ундан фойдаланиш мумкин.

Хосилавий тоифа маълумотларнинг асосий тоифалар кетма-кетлиги ва алмашиш буфери бошига нисбатан сурилишни билдирувчи бутун сонли қийматга эга элементлар тўплами билан характерланади. Сурилиш мусбат ёки манфий бўлиши мумкин ва бир биридан фарқ қилиши шарт эмас, хамда

тартибланганлиги талаб этилмайди. Шундай қилиб у асосий тоифасининг кетма-кетлигидан иборат бўлади ва маълумотларнинг бир элементи янги тоифада кўп марта учтариши мумкин.

MPI_TYPE_CONTIGUOUS(COUNT,TYPE,NEWTYPE,IERR)

INTEGER COUNT, TYPE,NEWTYPE,IERR

TYPE асосий тоифали кетма-кет COUNT та элементлардан иборат янги NEWTYPE маълумотлар тоифасини яратиш. Аслида яратилган маълумот тоифаси асосий тоифали маълумотлар массивига алохида объект сифатида қарилиши дейиш мумеин.

Навбатдаги мисолда newtype янги маълумотлар тоифаси яратилади ва ундан кейин кетма-кет бешта бутун сонни жўнатиш учун қўлланилади.

call MPI_TYPE_CONTIGUOUS(5,MPI_INTEGER,newtype,ierr)

MPI_TYPE_VECTOR(COUNT,BLOCKLEN,STRIDE,TYPE,NEWTYPE,IERR)

INTEGER COUNT,BLOCKLEN,STRIDE,TYPE,NEWTYPE,IERR

Маълумотларнинг асосий TYPE тоифали BLOCKLEN та элементидан иборат COUNT та блокдан ташкил топган янги NEWTYPE маълумот тоифасини яратиш. Навбатдаги блок олдинги блок бошидан кейинги асосий тоифанинг STRIDE та элементидан кейин бошланади.

Навбатдаги мисолда newtype янги маълумот тоифаси яратилади. Бу тоифа яратилгандан кейин ягона бутун объект сифатида қараладиган олтита маълумотлар элементини жўнатиш мумкин. Олтита маълумотлар элементни қуйидагича тасвирлаш мумкин (маълумот элемент тоифаси, жўнатиш буфери бошидан маълумотлар элемент миқдори):

{(MPI_REAL,0),(MPI_REAL,1),(MPI_REAL,2),(MPI_REAL,5),(MPI_REAL,6),(MPI_REAL,7)}

count=2

blocklen=3

stride=5

call MPI_TYPE_VECTOR(count,blocklen,stride,MPI_REAL,newtype,ierr)

MPI_TYPE_HVECTOR(COUNT,BLOCKLEN,STRIDE,TYPE,NEWTYPE, IERR)

INTEGER COUNT,BLOCKLEN,STRIDE,TYPE,NEWTYPE,IERR

Маълумотларнинг асосий TYPE тоифали BLOCKLEN та элементдан иборат COUNT та блокдан ташкил топган янги NEWTYPE маълумот тоифасини яратиш. Навбатдаги блок олдинги блок бошидан кейинги STRIDE байтдан кейин бошланади.

MPI_TYPE_INDEXED(COUNT,BLOCKLENS,DISPLS,TYPE,NEWTYPE, IERR)

INTEGER COUNT,BLOCKLENS(*),DISPLS(*),TYPE,NEWTYPE,IERR

Асосий тоифанинг BLOCKLENS(I) та элементидан иборат COUNT та блокдан ташкил топган янги NEWTYPE тоифасини яратиш. I инчи блок жўнатиш буфери бошидан бошлаб асосий тоифанинг DISPLS(I) та элементдан кейин бошланади. Хосил бўлган тоифани вектор тоифаси умумлашмаси сифатида қараш мумкин.

Навбатдаги мисолда double precision тоифали матрицани пастки учбурчагини сақлаш учун newtype маълумот тоифаси яратилади:

```
do i=1,n
   blocklens(i)=n-i+1
 displs(i)=n*(i-1)+i-1
end do
```

callMPI_TYPE_INDEXED(n,blocklens,displs,MPI_DOUBLE _PRECISION, newtype,ierr)

MPI_TYPE_HINDEXED(COUNT,BLOCKLENS,DISPLS,TYPE,NEWTYPE, IERR)

INTEGER COUNT,BLOCKLENS(*),DISPLS(*),TYPE,NEWTYPE,IERR

Маълумотлар асосий тоифанинг BLOCKLENS(I) та элементидан иборат

COUNT та блокдан ташлил топган NEWTYPE яхши тоифа яратиш. I инчи блок жўнатиш буфер бошидан DISPLS(I) байтдан бошланади.

MPI_TYPE_STRUCT(COUNT,BLOCKLENS,DISPLS,TYPES,NEWTYPE,
IERR)

INTEGER COUNT,BLOCKLENS(*),DISPLS(*),TYPES(*),NEWTYPE,IERR

TYPES(I) тоифали BLOCKLENS(I) та элементли COUNT блокдан ташкил топган NEWTYPE янги структурали тоифа яратиш. I инчи блок жўнатиш буфери бошидан хисоблаганда DISPLS(i) байтдан кейин бошланади.

{(MPI_DOUBLE_PRECISION,0),(MPI_DOUBLE_PRECISION,8),(MPI_DOUB
LE_PRECISION,16),(MPI_CHARACTER,24)

(MPI_CHARACTER,25)}

BLOCKLENS(1)=3

blocklens(2)=2

types(1)= MPI_DOUBLE_PRECISION

types(2)=MPI_CHARACTER

displs(1)=0

displs(2)=24

call MPI_TYPE_STRUCT(2,blocklens,displs,types,newtype,ierr)

MPI_TYPE_COMMIT(DATATYPE,IERR)

INTEGER DATATYPE,IERR

Яратилган хосилавий DATATYPE маълумот тоифасини қайт этилади. Қайт этгандан кейин жорий тоифани мавжуд асосий тоифалар билан бир қаторда маълумот алмашиш амалларида қўллаш мумкин бўлади. Мавжуд тоифаларни қайт этиш керак эмас.

MPI_TYPE_FREE(DATATYPE,IERR)

INTEGER DATATYPE,IERR

DATATYPE хосилавий тоифани йўқотиш. DATATYPE параметрида

MPI_DATATYPE_NULL қиймати ўрнатилади. Йўқотилаётган жорий, хосилавий тенг билан боғлиқ бошланган ихтиёрий амал нормал тугатилиши кафолатланади. Бунда хосилавий DATATYPE тоифали маълумотлар йўқолмайди ва кейинчалик улардан фойдаланиш мумкин. Мавжуд асосий тоифалар ўчирилмайди.

MPI_TYPE_SIZE(DATATYPE,SIZE,IERR)

INTEGER DATATYPE,SIZE,IERR

DATATYPE маълумотлар тоифасининг байтлардаги SIZE ўлчамини аниқлаш(жорий тоифанинг битта элементининг хотирада эгаллаган ўлчами)

MPI_ADDRESS(LOCATION,ADDRESS,IERR)

<type> LOCATION(*)

INTEGER ADDRESS,IERR

Компьютер тезкор хотирасида LOCATION массивининг ADDRESS жойлашиш манзили абсолют байтини аниқлаш. Манзил қиймати MPI_BOTTOM система константасида сақланувчи база манзили бошидан бошлаб хисобланади. Бу процедура Фортран тилида хам, ва Си тилида хам объектларни абсолют манзилини аниқлайди. Си тилида бунинг учун бошқа воситалар хам мавжуд. Си тилида ADDRESS параметри MPI_Aint тоифали.

Навбатдаги мисолда newtype маълумотларнинг янги тоифаси тавсифланади. У қайт этиш бўлгандан кейин бу тоифа double precision ва character(1) тоифали иккита маълумотлар элементини жоига элемент сифатида жўнатиш мумкин бўлади. Жўнатиш буфер манзили сифатида MPI_BOTTOM база манзили қўлланилади, dat1 ва dat2 маълумотлар элементларини сурилишини аниқлаш учун MPI_ADDRESS процедураси чақирилади. Жўнатишдан олдин янги тоифа MPI_TYPE_COMMIT процедураси чақирилиб қайт қилинади.

blocklens(1)=1

blocklens(2)=1

types(1)=MPI_DOUBLE_PRECISION

types(2)=MPI_CHARACTER

call MPI_ADDRESS(dat1,address(1),ierr)

displs(1)=address(1)

call MPI_ADDRESS(dat2,address(2),ierr)

displs(2)=address(2)

call MPI_TYPE_STRUCT(2,blocklens,displs,types,newtype,ierr)

call MPI_TYPE_COMMIT(newtype,ierr)

callMPI_SEND(MPI_BOTTOM,1,newtype,dest,tag,MPI_COMM_WORLD, IERR)

MPI_TYPE_LB(DATATYPE,DISPL,IERR)

INTEGER DATATYPE,DISPL,IERR

DATATYPE тоифали маълумот элементининг пастки (қуйи) чегарасини маълумотлар буфери бошига нисбатан DISPL сурилишини байтларда аниқлаш.

MPI_TYPE_UB(DATATYPE,DISPL,IERR)

INTEGER DATATYPE,DISPL,IERR

DATATYPE тоифали маълумот элементининг юқори чегарасини маълумотлар буфери бошига нисбатан DISPL сурилишини байтларда аниқлаш.

MPI_TYPE_EXTENT(DATATYPE,EXTENT,IERR)

INTEGER DATATYPE,EXTENT,IERR

DATATYPE тоифали маълумотларнинг EXTENT диопозонини (жорий тоифали маълумот элементининг юқори ва қуйи чегараси орасидаги фарқни) аниқлаш.

Навбатдаги мисолда маълумотларнинг хосилавий тоифаси матрица устунларини қайта жойлаш учун ишлатилади. MPI_TYPE_VECTOR процедураси ёрдамида яратилган matr_rev маълумотлар тоифаси жорий

жараёнда устунлари тескари тартибда қўйиб чиқилган матрицанинг локал қисмини ташкил этади. Қайт этилгандан кейин бу тоифа маълумотлар алмашишда қўлланилади. Агар матрица ўлчами N дастурдаги жараёнларнинг сонига бутун сонли бўлинса дастур тўғри ишлайди.

```
program example18
include 'mpif.h'
integer ierr,rank,size,N,n1
parametr (N=8)
double precision a(N,N),b(N,N)
call MPI_INIT(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD,size,ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)
n1=(N-1)/size+1
call work(a,b,N,n1,size,rank)
call MPI_FINALIZE(ierr)
end
```

Маълумотларни ўраш ёки ихчамлаш.

Хар хил тоифали маълумотларни алмашиш учун хосилавий тоифаларни яратиш усули билан бирга маълумотларни ўраш ва очиш усули қўлланилади. Хар хил тоифали ёки хотиранинг кетма-кет ячейкаларида жойлашмаган маълумотлар битта узлуксиз буферга жойланади, жўнатилади ва олинган хабар яна хотиранинг керакли ячейкалари бўйлаб тақсимланади.

```
MPI_PACK(INBUF,INCOUNT,DATATYPE,OUTBUF,OUTSIZE,POSITION,C
OMM,IERR)
<type>INBUF(*),OUTBUF(*)
INTEGER INCOUNT,DATATYPE,OUTSIZE,POSITION,COMM,IERR
```

INBUF массивидаги DATATYPE тоифали INCOUNT та элементни массив бошига нисбатан POSITION байтга силжитиб OUTBUF массивига

ўраш. OUTBUF буфери камида OUTSIZE байт ўлчамга эга. Процедура бажарилгандан кейин POSITION параметри ёзув ўлчамига тенг байтга ошади. COMM параметри кейинчалик хабар жўнатиладиган коммуникаторни билдиради. Ўралган маълумотларни жўнатиш учун маълумотларнинг MPI_PACKED тоифаси қўлланилади.

MPI_UNPACK(INBUF,INSIZE,POSITION,OUTBUF,OUTCOUNT,
DATATYPE,COMM,IERR)

<type>INBUF(*),OUTBUF(*)

INTEGER INSIZE,POSITION,OUTCOUNT,DATATYPE,COMM,IERR

INBUF массивидаги DATATYPE тоифали OUTCOUNT элементни массив бошига нисбатан POSITION байт суриб OUTBUF массивга ўраш. INBUF массиви ўлчами INSIZE байтдан кам эмас.

MPI_PACK_SIZE(INCOUNT,DATATYPE,COMM,SIZE,IERR)

INTEGER INCOUNT,DATATYPE,COMM,SIZE,IERR

DATATYPE тоифали INCOUNT та элементни ўраш учун зарур SIZE хотира хажмини аниқлаш. Ўраш учун зарур ўлчам ўралиётган маълумот элементлари ўлчамлари йиғиндисидан ошиб кетиши мумкин.

Навбатдаги мисолда buf массиви real тоифали a массивнинг 10 та элементини ва character тоифали b массивнинг 10 та элементини ўрашда буфер сифатида ишлатилади. Хосил бўлган хабар MPI_BCAST процедураси ёрдамида 0 инчи жараёндан қолган жараёнларга жўнатилади. Қабул қилинган хабар MPI_UNPACK процедурасини чақириш орқали очилади.

```
program example19
include 'mpif.h'
integer ierr,rank,position
real a(10)
character b(10),buf(100)
call MPI_INIT(ierr)
```

```
call MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)
do i=1,10
a(i)=rank+1.0
if(rank .eq.0) then
b(i)='a'
else
b(i)='b'
end if
end do
position=0
if(rank .eq.0) then
callMPI_PACK(a,10,MPI_REAL,buf,100,position, MPI_COMM_WORLD,ierr)
callMPI_PACK(b,10,MPI_CHARACTER,buf,100,position,
MPI_COMM_WORLD,ierr)
call MPI_BCAST(buf,100, MPI_PACKED,0, MPI_COMM_WORLD,ierr)
else
call MPI_BCAST(buf,100, MPI_PACKED,0, MPI_COMM_WORLD,ierr)
position=0
callMPI_UNPACK(buf,100,position,a,10,MPI_REAL,
MPI_COMM_WORLD,ierr)
callMPI_UNPACK(buf,100,position,b,10,
MPI_CHARACTER,MPI_COMM_WORLD,ierr)
end if
print *,'process',rank,'a=', a,'b=', b
call MPI_FINALIZE(ierr)
end.
```