MI-SPOL-13 (MI-PDP.16) (přednáška 7 a 9)

Programový model nad distribuovanou pamětí: MPI

(procesy, komunikátory, 2-bodové a skupinové komunikační operace, blokující a neblokující operace a jejich komunikační módy)

Hybridní MPI + OpenMP model

MPI - Message passing interface → rozhraní zpráv mezi procesy paralelního programu
- pracuje nad distribuovanou pamětí (NUMA)
- každý proces je vždy součástí alespoň jedné skupiny (číslování od 0 do #-1)
  - v každé skupině obecně jiné číslo

Komunikátory - součástí (parametr) každé komunikační operace
- určují v rámci které skupiny probíhá komunikace - MPI_COMM_WORLD pro všechny procesy
- index (uvnitř skupiny) a index (mezi skupinami)

MPI_Comm_rank - číslo procesu v dané skupině
MPI_Comm_size - počet procesů dané skupiny

Komunikační operace
- 2 bodové - mezi dvěma procesy
- kolektivní - všechny s daným komunikátorem

- blokující - čeká se na splnění určité podmínky
- neblokující - hned ukončí - musí ohlídat později

MPI_Send (*buff, count, MPI_Datatype, dest, tag, comm)
MPI_Recv (*buff, count, MPI_Datatype, src, tag, comm, &status)

dest: cílový proces
status: struktura obsahující source a tag přijaté zprávy [MPI_STATUS_IGNORE]
MPI_Datatype: typ daných přenášených dat [MPI_INT, MPI_DOUBLE, ...]
tag: značka přenášených dat [MPI_ANY_TAG]
src: od koho mám přijímat [MPI_ANY_SOURCE]

- ze základu blokující operace (= buffered nebo synchronous mode)

MPI_Bsend - buffered → uloží do systémového bufferu (nevčká na cíl)
MPI_Ssend - synchronous → končí až když cílový proces inicioval příjem
MPI_Rsend - ready mode → příjem už musí být inicován, jinak konec s chybou

- všechny operace mají i neblokující variantu (MPI_Isend, MPI_Ibcend, ...)
  - musíme explicitně testovat dokončení operace
  - buffer do té doby nelze modifikovat
- funkce mají dodatečný parametr MPI_Request
  → MPI_Test, MPI_Wait & ... any, ... all pro hromadné operace
- spojení MPI_Sendrecv
- MPI_Probe - kontroluje příchod zprávy

Kolektivní komunikační operace

1:N
- → one-to-all broadcast - MPI_Bcast (stejný multicast)
- → one-to-all scatter - MPI_Scatter
- → all-to-one gather - MPI_Gather - sbírá data ode všech
  - poslední dvě se liší pouze ve směru, latence jsou stejné

N:N
- → all-to-all broadcast
  = all-to-all gather - MPI_AllGather
- → all-to-all scatter - MPI_Alltoall

Hybridní model MPI + OpenMP

- v rámci jednoho výpočetního uzlu / procesoru běží jeden nebo pár MPI procesů, každý se dělí na několik vláken pomocí OpenMP

- 1. proces na jeden výpočetní uzel → pořád jeden na celý uzel
- 2. proces na každý procesor (tj. socket) → lepší přístup ke sdílené paměti

- musí se inicializovat pomocí MPI_Init_thread na požadovanou míru spolupráce

MPI_THREAD_SINGLE - nedělíme se na vlákna
MPI_THREAD_FUNNELED - pouze hlavní vlákno může volat MPI funkce
MPI_THREAD_SERIALIZED - MPI funkce smí v jednu chvíli volat jen jedno vlákno
MPI_THREAD_MULTIPLE - víceprocesový model