Гибридное программирование MPI+OpenMP

Александр Позднеев
Аспирант 2-го года кафедры АНИ
pozdneev@gmail.com

Факультет вычислительной математики и кибернетики
Московский государственный университет имени М. В. Ломоносова

Специальный курс «Терафлопные вычисления»

План

Идеология

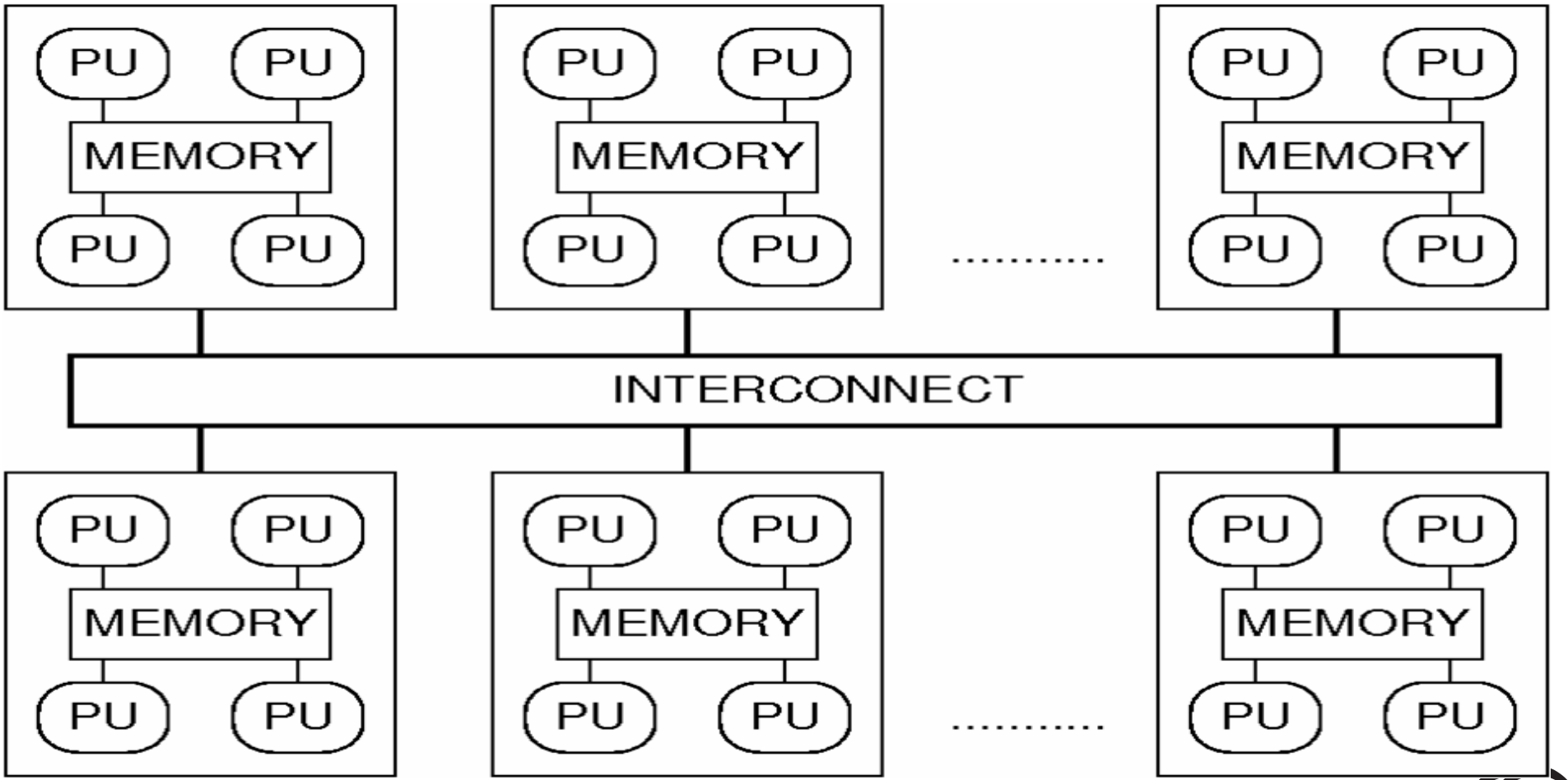
- Каждый MPI-процесс порождает несколько нитей:
  - OpenMP
  - POSIX threads (Pthreads)
  - присущие данной системе нити

Мотивация
Актуальность

- Широкое распространение распределенных систем с многоядерными узлами, кластеров из SMP-систем
  - IBM System Cluster 1600 (IBM eServer pSeries 690 Regatta)
  - Blue Gene/P
  - Суперкомпьютер СКИФ МГУ
  - IBM eServer pSeries 690 Regatta
- Эффективное использование многоуровневой иерархии памяти
  - Динамическая балансировка: OpenMP против MPI
  - Обращение к общей памяти вместо MPI-сообщений
  - Меньший объем вычислений:
    - ★ Например, MPI-процессы обмениваются данными $\mathbf{A}_1$ и $\mathbf{A}_2$, потом каждый из них выполняет операцию $\mathbf{F}(\mathbf{A}_1, \mathbf{A}_2)$
- Совмещение вычислений, межпроцессных обменов, ввода/вывода

Мотивация

Распределенная система с SMP-узлами



Необходимое ПО

- Компилятор с поддержкой MPI и нитей:
  - `mpif90_r`, `mpicc_r`, `mpixlf90_r`, `mpixlc_r`
  - Blue Gene/P: `bgxlf_r`, `bgxlc_r`
  - `-qsmp=omp`
- Планировщик, способный «правильно» разместить задачу:
  - Portable Batch System (PBS)
  - IBM Parallel Operating Environment (POE)
  - IBM LoadLeveler

«Правильная» инициализация MPI с поддержкой нитей

```
int main(int argc, char **argv) {
  int required = MPI_THREAD_FUNNELED;
  int mpi_rank, mpi_size, mpi_err, provided;
  MPI_Comm comm = MPI_COMM_WORLD;

  mpi_err = MPI_Init_thread(&argc, &argv, required, &provided);
  mpi_err = MPI_Comm_rank(comm, &mpi_rank);

  if (mpi_rank == 0) {
    switch (provided) {
      case MPI_THREAD_SINGLE: /* */ break;
      case MPI_THREAD_FUNNELED: /* */ break;
      case MPI_THREAD_SERIALIZED: /* */ break;
      case MPI_THREAD_MULTIPLE: /* */ break;
      default: /* */ break;
    }
  }
```

Уровни поддержки нитей

- MPI THREAD SINGLE
  - MPI-процесс исполняет единственную нить
- MPI THREAD FUNNELED
  - MPI-процесс может быть многонитевым, но делать MPI-вызовы разрешено только тому процессу, который проводил инициализацию MPI
- MPI THREAD SERIALIZED
  - MPI-процесс может быть нитевым, но в данный момент времени MPI-вызов делает лишь одна нить
- MPI THREAD MULTIPLE
  - MPI-процесс может быть многонитевым, и несколько нитей могут вызывать MPI-функции одновременно

Замечания

- Вызов `MPI_INIT` имеет тот же эффект, что и обращение к `MPI_INIT_THREAD` с аргументом `required = MPI_THREAD_SINGLE`
- Использование нитей при `MPI_THREAD_SINGLE` может привести к непредсказуемым проблемам
- Номера процессов в функциях посылки и приема идентифицируют MPI-процесс, но не нить. Сообщение, посланное MPI-процессу может быть получено любой нитью этого процесса

Пример командного файла для системы PBS

```
#!/bin/sh
#PBS -lnodes=3:ppn=1
#PBS -lwalltime=0:15:00
module load intel-compilers
module load intel-mpich-ib
export OMP_NUM_THREADS=2
cd $PBS_O_WORKDIR
mpiexec ok
```

Пример командного файла для системы POE

```
#!/bin/sh
# @ notification = never
# @ output = $(jobid).out
# @ error = $(jobid).err
# @ job_type = parallel
# @ network.mpi = csss,shared,us
# @ wall_clock_limit = 00:15:00
# @ requirements = (Pool==2)
# @ node = 2
# @ tasks_per_node = 1
# @ resources = ConsumableCpus(4)
#
# @ queue
export MP_EUILIB=us
export OMP_NUM_THREADS=4
./ok
```

Пример командного файла для Loadleveler (Regatta)

```
#!/bin/bash
#@ output = $(executable).$(jobid).$(stepid).out
#@ error = $(executable).$(jobid).$(stepid).err
#@ notification  = never
#@ job_type = parallel
#@ node = 1
#@ tasks_per_node = 4
#@ node_usage = not_shared
#@ resources = ConsumableCpus(3)
#@ wall_clock_limit = 00:05:00
#@ environment = COPY_ALL; \
                 OMP_NUM_THREADS=3; AIXTHREAD_SCOPE=S; \
                 MEMORY_AFFINITY=MCM; \
                 MP_SHARED_MEMORY=yes; MP_WAIT_MODE=poll; \
                 MP_SINGLE_THREAD=yes; MP_TASK_AFFINITY=MCM
#@ queue
/usr/local/bin/mpirun -np 4 ok
```