Суперкомпьютерные вычислительные технологии.

Лекционно-практический курс
для студентов 5 курса факультета ВМиК МГУ
сентябрь – декабрь 2013 г.

Лектор доцент Н.Н.Попова

Лекция 7

18 октября 2013 г.

Тема

- Комментарии по заданию 2
- Архитектура и ПО BlueGene/P

Задание 2.

Срок: 15 ноября 2013

Численное решение задачи Дирихле.

- Метод SOR.
- Разработка параллельной MPI-программы и исследование ее эффективности.
- Параметры, передаваемые в командной строке:

  - Первый параметр: m – число точек по одному измерению для задания двумерной сетки. По умолчанию – 512.

  - Второй параметр — точность. По умолчанию – 0.01.

Пример параллельной программы. Метод Якоби (1)

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include "mpi.h"
#define CHAR char
#define REAL double
#define INT  int
#define OUTPUT stdout       /* output to standard out                    */
#define PLOT_FILE "plots"   /* output files base name                    */
#define INCREMENT 500       /* number of steps between convergence check  */
#define P 1                 /* define processor count for serial codes    */
#define K 0                 /* current thread number for serial code is 0 */
#define MAX_M 4096          /* maximum size of indices of Array u         */
#define M_DEFAULT 512       /* default size of indices of array u */
#define MAXSTEPS 500000     /* Maximum number of iterations               */
#define EPS     0.01        /* Numerical Tolerance */
#define EPS_DEFAULT 0.01    /* default accuracy */
#define PI 3.14159265       /* pi */

#define FILE_OUT 0          /* Output results to the file
```

Пример параллельной программы. Метод Якоби (2)

```
/* begin function prototyping  */
REAL **allocate_2D(int m, int n);
REAL my_max(REAL a, REAL b);
void init_array( INT m, INT n, REAL **a);
void bc( INT m, INT n, REAL **a, INT k, INT p );
INT write_file( INT m, INT n, REAL **u, INT k, INT p );
INT update_jacobi( INT m, INT n, REAL **u, REAL **unew, REAL *gdel);
INT replicate( INT m, INT n, REAL **u, REAL **ut );
INT transpose( INT m, INT n, REAL **u, REAL **ut );
void neighbors(INT k, INT p, INT UNDEFINED, INT *below, INT *above);
INT update_bc_2( INT mp, INT m, REAL **vt, INT k, INT below, INT
above );
/* end function prototyping */
```

Пример параллельной программы. Метод Якоби (3)

```
INT main(INT argc, CHAR *argv[]) {
/********** MAIN PROGRAM ********************************
 * Solve Laplace equation using Jacobi iteration method  *
       *
 *******************************************************/
     INT m = M_DEFAULT, mp, k, p, below, above;
     long iter=MAXSTEPS;
     REAL TOL=EPS_DEFAULT, del, gdel,start,finish,time;
     CHAR line[80];
     REAL **v, **vt, **vnew;
     MPI_Init(&argc, &argv);                    /* starts MPI */
     MPI_Comm_rank(MPI_COMM_WORLD, &k); /* get current process id */
     MPI_Comm_size(MPI_COMM_WORLD, &p); /* get # procs from env or */
     if(k == 0) {
          fprintf(OUTPUT," Number of args in command line, argc :\n");

          fprintf(OUTPUT," argc = %d :\n", argc);
          if (argc >= 2)
             fprintf(OUTPUT," arg[1]= %s :\n", argv[1]);
          if (argc >= 3)
```

Пример параллельной программы. Метод Якоби (4)

```
MPI_Bcast(&m, 1, MPI_INT, 0, MPI_COMM_WORLD);
MPI_Bcast(&TOL, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);

mp = m/p;

v  = allocate_2D(m, mp);  /* allocate mem for 2D array */
vt = allocate_2D(mp, m);
vnew = allocate_2D(mp, m);
gdel = 1.0;
iter = 0;
bc(m, mp, v, k, p); /* initialize and define B.C. for v */
transpose(m, mp, v, vt);  /* solve for vt */
                    /* driven by need of update_bc_2 */
replicate(mp, m, vt, vnew);          /* vnew = vt */
neighbors(k, p, MPI_PROC_NULL, &below, &above);   /* domain borders */
```

Пример параллельной программы. Метод Якоби (5)

```
while (gdel > TOL) {  /* iterate until error below threshold */
      iter++;           /* increment iteration counter */
      if(iter > MAXSTEPS) {
        fprintf(OUTPUT,"Iteration terminated (exceeds %6d", MAXSTEPS);
        fprintf(OUTPUT," )\n");
        return (0);       /* nonconvergent solution */
      }
/* compute new solution according to the Jacobi scheme */
      update_jacobi(mp, m, vt, vnew, &del);  /* compute new vt */
      if(iter%INCREMENT == 0) {
        MPI_Allreduce( &del, &gdel, 1, MPI_DOUBLE,
            MPI_MAX, MPI_COMM_WORLD );  /* find global max error */
        if( k == 0) {
          fprintf(OUTPUT,"iter,del,gdel: %6d, %lf %lf\n",iter,del,gdel);
        } }
      update_bc_2( mp, m, vt, k, below, above); /* update b.c. */
    }
```

Пример параллельной программы. Метод Якоби (6)

```
        if (k == 0) {
              finish=MPI_Wtime();
          time=start+finish;
          fprintf(OUTPUT,"Stopped at iteration %d\n",iter);
          fprintf(OUTPUT,"The maximum error = %f\n",gdel);
          fprintf(OUTPUT,"Time = %f\n",time);
        }
        if (FILE_OUT) {
              /* write v to file for use in MATLAB plots */
              transpose(mp, m, vt, v);
              write_file( m, mp, v, k, p );
              MPI_Barrier(MPI_COMM_WORLD);
        }
        free(v); free(vt); free(vnew); /* release allocated arrays  */
             return (0);
    }
```

Пример параллельной программы. Метод Якоби (7)

```
INT update_bc_2( INT mp, INT m, REAL **vt, INT k, INT below, INT above ) {
   MPI_Status status[6];
   MPI_Sendrecv( vt[mp  ]+1, m, MPI_DOUBLE, above, 0,
                 vt[0   ]+1, m, MPI_DOUBLE, below, 0,
                 MPI_COMM_WORLD, status );
   MPI_Sendrecv( vt[1   ]+1, m, MPI_DOUBLE, below, 1,
                 vt[mp+1]+1, m, MPI_DOUBLE, above, 1,
                 MPI_COMM_WORLD, status );
   return (0);
}
```

Пример параллельной программы. Метод Якоби (8)

```
void bc(INT m, INT n, REAL **u, INT k, INT p) {
/************ Boundary Conditions ****************************************
 *  PDE: Laplacian u = 0;      0<=x<=1;  0<=y<=1                          *
 *  B.C.: u(x,0)=sin(pi*x); u(x,1)=sin(pi*x)*exp(-pi); u(0,y)=u(1,y)=0   *
 *  SOLUTION: u(x,y)=sin(pi*x)*exp(-pi*y)                                *
 ************************************************************************/
INT i;
   init_array( m, n, u);                    /* initialize u to 0 */
 if (p > 1) {
   if (k == 0) {
     for (i = 0; i <=m+1; i++) {
       u[i][0] = sin(PI*i/(m+1));           /* at y = 0; all x */
     }
   }
/*            ПРОДОЛЖИТЬ                                                  **/
```

Пример параллельной программы. Метод Якоби (8)

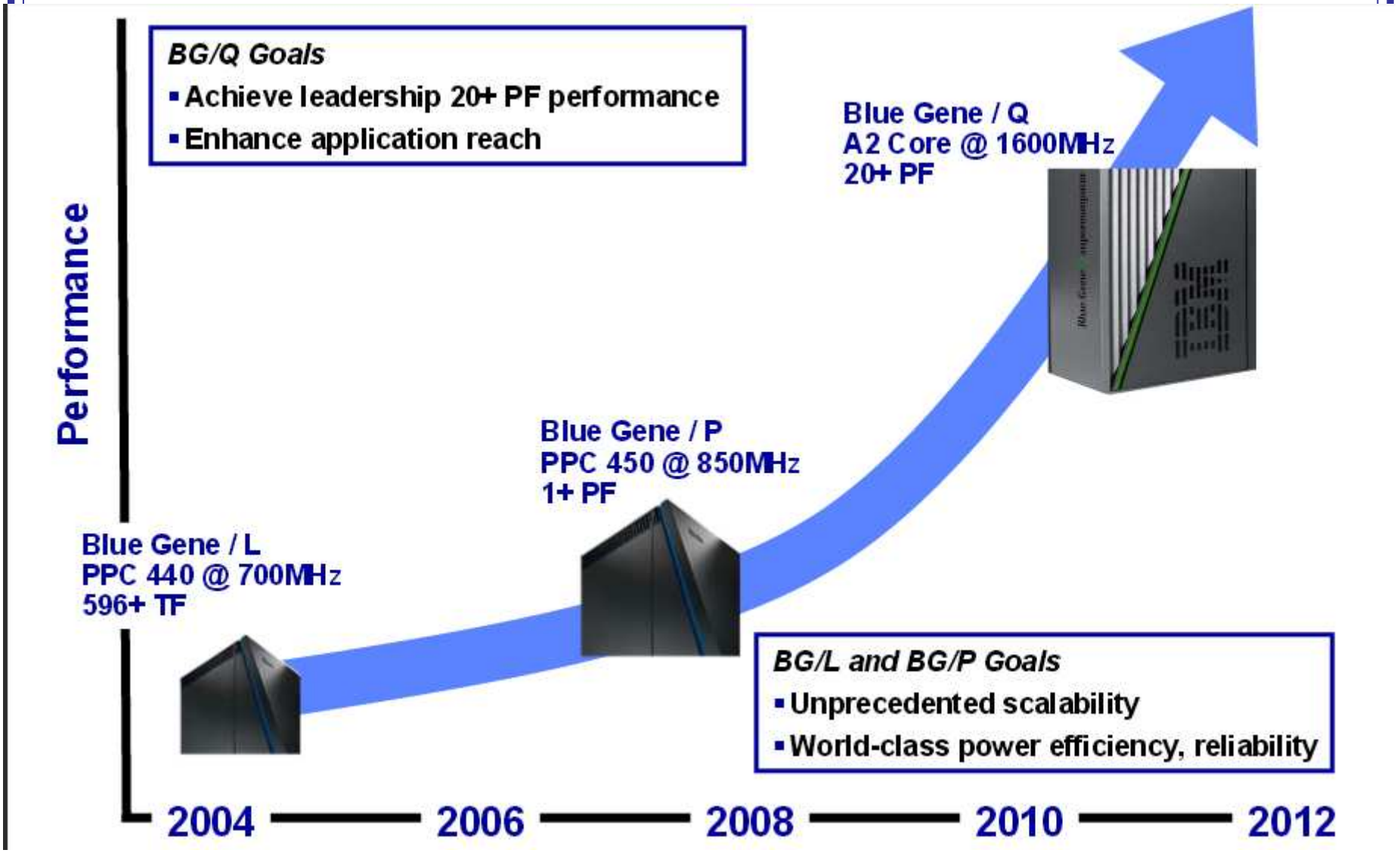
```
INT update_jacobi( INT m, INT n, REAL **u, REAL **unew, REAL *del) {
/*****************************************************************
 * Updates u according to Jacobi method                          *
 * m        - (INPUT)  size of interior rows                     *
 * n        - (INPUT)  size of interior columns                  *
 * u        - (INPUT)  solution array                            *
 * unew     - (INPUT)  next solution array                       *
 * del      - (OUTPUT) error norm between 2 solution steps       *

 *****************************************************************/
```

Архитектура и программное обеспечение Blue Gene

Проект Blue Gene

- ***Blue Gene/L***
  - Начинался как массивно-параллельный компьютер для изучения фолдинга белков
  - Первый прототип - 2004 г. Первая строка в Top 500 с производительностью в 70.72 Тфлопс/с
  - 2-х ядерный чип
- ***Blue Gene/P***
  - Продолжение линейки Blue Gene
  - Увеличена частота процессора и объем памяти
  - 4-х ядерный чип (технология system-on-a –chip)
  - Самая большая система на основе Blue Gene/P установлена в Германии (JUGENE)
  - ***Blue Gene/Q***
  - 2012 год, производительность ~20 Пфлопс/с
  - 16-ядерный чип

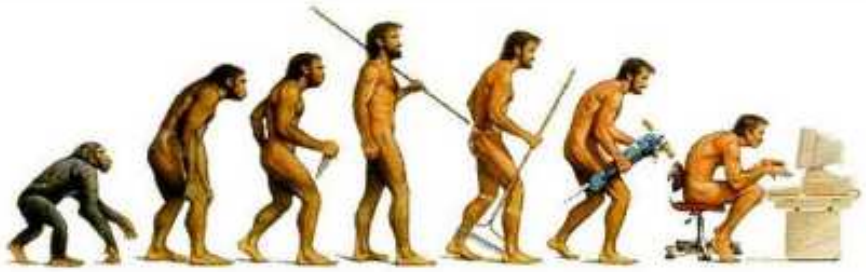
Blue Gene Road map



Общая характеристика систем Blue Gene

- Массивно-параллельные системы с распределенной памятью
- Технология System-on-chip (4 ядра, 8 FPU, контроллер памяти и др. на одном ASIC)
- Высокая плотность упаковки
  - процессоры с низким энергопотреблением
- Высокопроизводительный интерконект
  - несколько коммуникационных подсистем для различных целей
- Ультра легкая ОС
  - выполнение вычислений и ничего лишнего
- Стандартное ПО
  - Fortran/C/C++ и MPI

Blue Gene Evolution



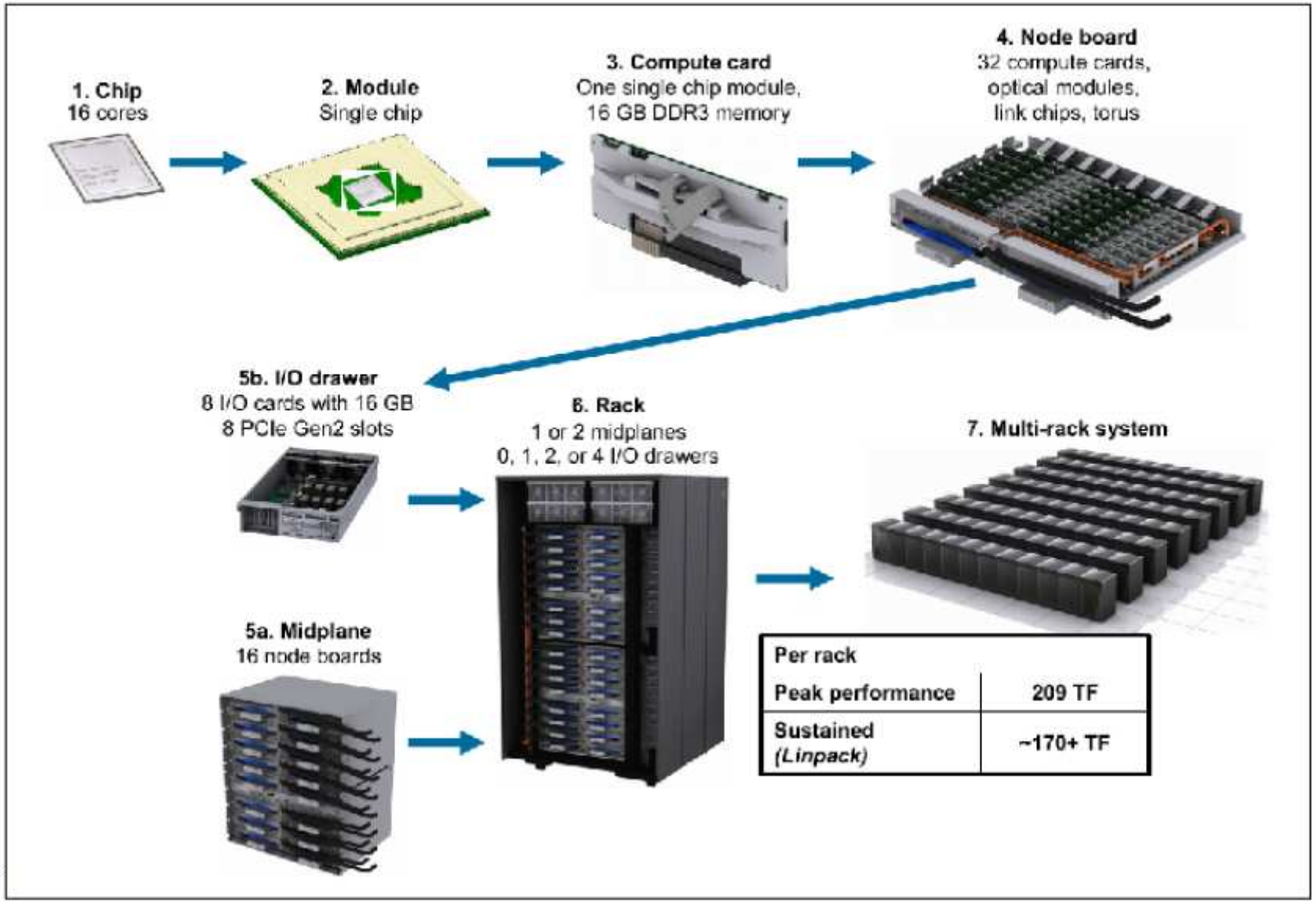
- BG/L (5.7 TF/rack) – 130nm ASIC (2004 GA)
  - Embedded 440 core, dual-core system-on-chip (SOC)
  - Memory: 0.5/1 GB/node
  - Largest BG/L: LLNL @ 104 racks with 212,992 cores = 0.6 PF
- BG/P (13.9 TF/rack) – 90nm ASIC (2007 GA)
  - Embedded 450 core, quad-core SOC
  - Memory: 2/4 GB/node, DMA
  - SMP support, OpenMP, MPI
  - Largest BG/P: FZ Jülich @ 72 racks with 294,912 cores = 1 PF
- BG/Q (209 TF/rack) – 45nm ASIC+ (2012 GA)
  - A2 core, 16 core/64 thread SOC
  - 16 GB/node
  - Speculative execution, sophisticated L1 prefetch, transactional memory, fast thread handoff, compute + IO systems
  - Largest BG/Q: LLNL @ 96 racks with 1,572,864 cores = 20 PF

Blue Gene Characteristics

	BG/L	BG/P	BG/Q
Compute Nodes			
Processor	32-bit PowerPC 440	32-bit PowerPC 450	**64-bit PowerPC (A2 Core)**
Processor Frequency	700 MHz	850 MHz	**1.6 GHz**
Cores	2	4	**16**
Peak Performance (per Node)	5.6 GF	13.6 GF	**204.8 GF**
Coherency	Software Managed	SMP	**SMP + Speculation**
L1 Cache (per Core)	32 KB	32 KB	**16/32 KB**
L2 Cache (prefetch per Core/Thread)	14 stream	14 stream	**16 stream, List-based**
L3 Cache size (shared, per Node)	4 MB	8 MB	**32 MB**
Main Store/Node *(same for I/O Node)*	512 MB or 1 GB	2 GB or 4 GB	**16 GB**
Main Store Bandwidth	5.6 GB/s (16B wide)	13.6 GB/s (2*16B wide)	**43 GB/s**
Torus Network			
Topology	3D	3D	**5D**
Bandwidth	6*2*175 MB/s = 2.1 GB/s	6*2*425 MB/s = 5.1 GB/s	**32 GB/s**
Hardware Latency *(Nearest Neighbor)*	200 ns (32B packet) 1.6 µs (256B packet)	100 ns (32B packet) 800 ns (256B packet)	80 ns (32B packet) 640 ns (256B packet)
Hardware Latency *(Worst Case)*	6.4 µs (64 hops)	5.5 µs (64 hops)	**3 µs**
Per Rack			
Peak Performance	5.7 TF	13.9 TF	**209 TF**
Sustained Performance *(Linpack)*	4.6 TF	11.9 TF	**~170+ TF**
Power	~20 kW	~32 kW	**~100 kW**
Power Efficiency	0.23 GF/W	0.37 GF/W	**2.1 GF/W**

Blue Gene/P Hardware

Blue Gene/Q


1. Chip
16 cores
2. Module
Single chip
3. Compute card
One single chip module,
16 GB DDR3 memory
4. Node board
32 compute cards,
optical modules,
link chips, torus
5b. I/O drawer
8 I/O cards with 16 GB
8 PCIe Gen2 slots
6. Rack
1 or 2 midplanes
0, 1, 2, or 4 I/O drawers
7. Multi-rack system
5a. Midplane
16 node boards
Per rack
Peak performance
209 TF
Sustained
(Linpack)
~170+ TF

Blue Gene P

1 стойка

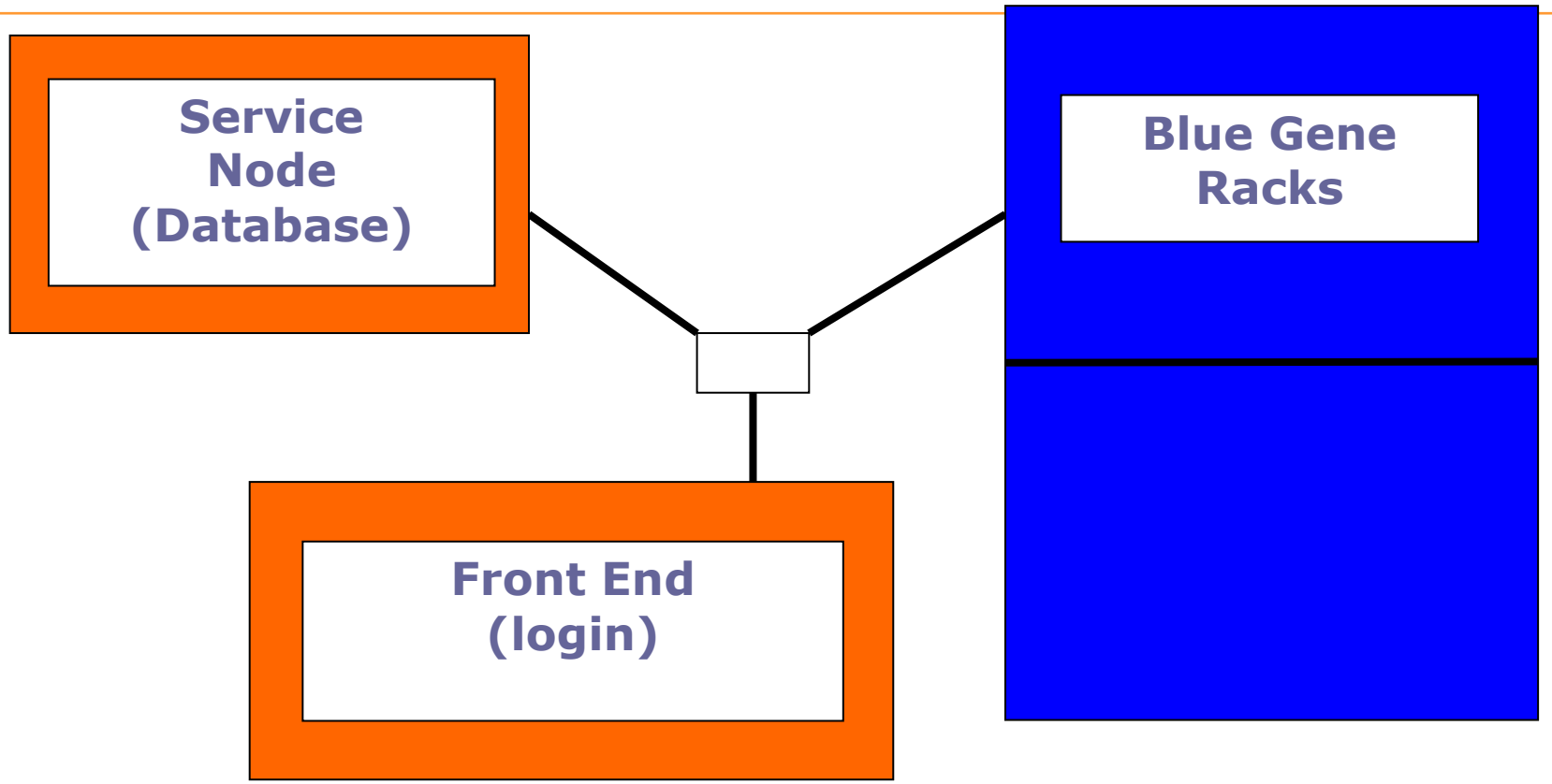
- 1024 четырехъядерных вычислительных узлов
- производительность одного вычислительного узла – 13.6 GF/s
- производительность 1 стойки– 13.9 Tflops
- оперативная память одного узла – 2 GB
- суммарная оперативная память в стойке– 2 TB
- узлов ввода/вывода 8 – 64
- Размеры - 1.22 x 0.96 x 1.96
- занимаемая площадь 1.17 кв.м.
- энергопотребление (1 стойка) - 40 kW (max)

Конфигурация BlueGene P факультета ВМиК

http://hpc.cs.msu.ru

- пиковая производительность 27.8 Tflop/s
- 2 стойки
- 2048 4-ех ядерных узлов
- общий объем ОЗУ 4 ТВ

BlueGene P



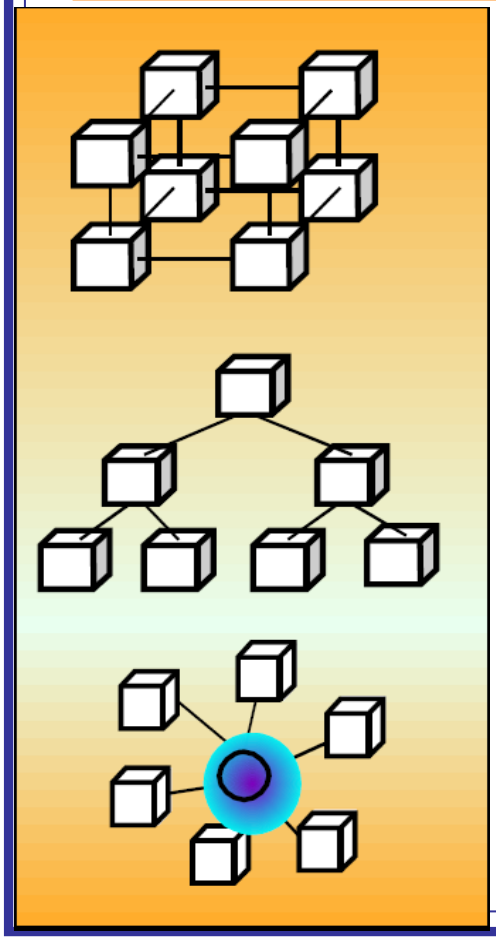
Компоненты Blue Gene P

- Основная единица – четырехядерный вычислительный узел (процессор) , ядро – PowerPC 450 850Mhz + память (2GB)
- Node card = 32 вычислительных узла + до 2х узлов ввода-вывода
- Стойка – 32 node cards
- Число процессоров в стойке
- Итоговое число ядер на стойку - 4096

Power PC 450 процессор

- 32-bit архитектура, 850 MHz
- integer unit
- load/store unit
- Специальное устройство double floating-point unit (dfpu)

Коммуникационные сети BGP



- Каждый вычислительный узел подключается к нескольким сетям:
  - 6 выходов в сеть, объединяющую вычислительные узлы в **трёхмерный тор**
  - 3 выхода в сеть для обмена коллективными сообщениями и связи с узлами ввода-вывода:

    коллективная сеть - дерево
  - 4 выхода в **высокоскоростную сеть** для обмена прерываниями
  - 1 выход в управляющую сеть

Процессоры ввода-вывода

Отличия по сравнению с вычислительным узлом:

- Установлена полноценная ОС
- Отсутствует подключение к сети тору
- Имеется выход в 10-гигабитную сеть Ethernet

Память

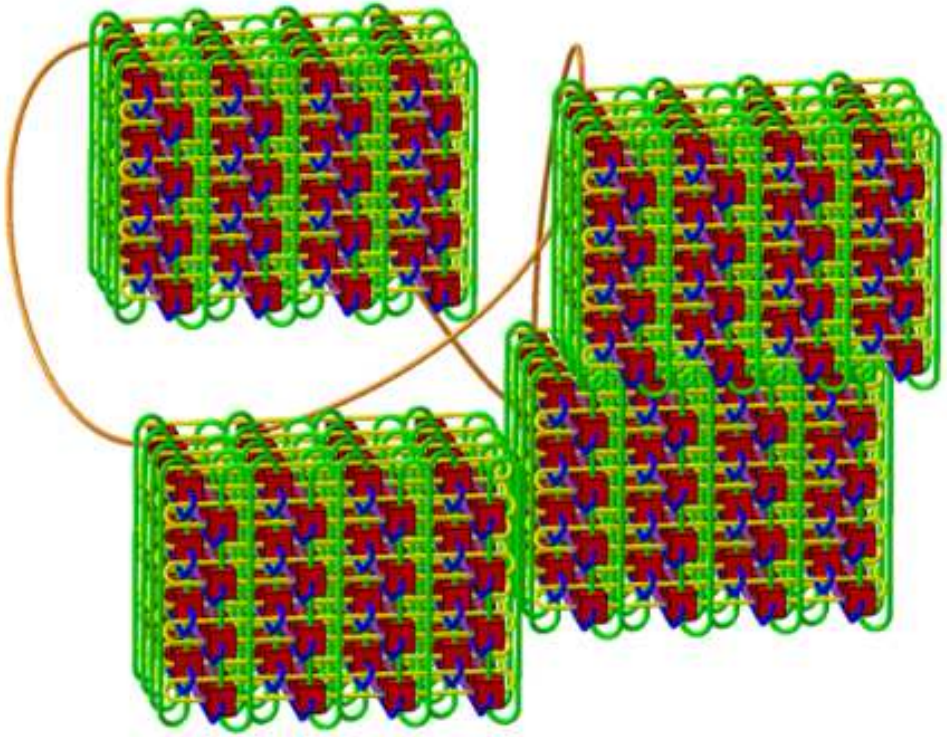
- Оперативная память – до 2GB на вычислительный узел, пропускная способность 13.6GBps
- Трёхуровневый кэш:
  - L1 – отдельный для каждого ядра, размер 32Kb
  - L2 – отдельный для каждого ядра, используется для предварительной выборки информации в кэш L1.
  - L3 – разделен на две части по 4MB, доступ к ним имеют все четыре ядра, для каждого есть канал чтения и канал записи. Связан с 10-гигабитной сетью (в том случае, если на карте имеется узел ввода-вывода)

Blue Gene/Q: innovative features

- PowerPC A2 processor
  - 16 computing cores + 1 “OS” core
  - each core: 4-way simultaneous multithreading (SMT)
  - 4-way double, precision vector floating-point unit (QPX) – 8 flops per cycle
  - two instructions per two threads per cycle: FP- and “integer”-instruction
- Memory
  - L1 intelligent prefetcher
  - 2 MB/core shared L2 cache
  - multiversion L2 cache
    - transactional memory
    - speculative execution
- Inteconnect
  - 5D torus

Scalability

Inter-Processor Communication



Network Performance

- All-to-all: 97% of peak
- Bisection: > 93% of peak
- Nearest-neighbor: 98% of peak
- Collective: FP reductions at 94.6% of peak

- **Integrated 5D torus**
  - Hardware assisted collective and barrier
  - FP addition support in network
  - Virtual Cut Through
  - RDMA direct from application
  - Wrapped
- **2 GB/s bandwidth on all 10 links (4 GB/s bidi)**
- **5D nearest neighbor exchange measured at ~1.75 GB/s per link**
- **Hardware latency**
  - Nearest: 80ns
  - Farthest: 3us (96-rack 20PF system)

Состав ПО

- Linux® на узлах ввода\вывода
- MPI (MPICH2) и OpenMP (2.5)
- Стандартное семейство компиляторов IBM XL: XLC/C++, XLF
- Компиляторы GNU
- Система управления заданиями LoadLeveler
- Файловая система GPFS
- Инженерная и научная библиотека подпрограмм (ESSL), математическая библиотека (MASS)

ОС вычислительного узла BlueGene P

- Compute Node Kernel (CNK)
  - “linux-подобная” ОС
  - Нет некоторых системных вызово (fork() в основном). Ограниченная поддержка mmap(), execve()
  - Минимальное ядро – обработка сигналов, передача системных вызовов к узлам ввода-вывода, старт-завершение задач, поддержка нитей
  - Большинство приложений, которые работают под Linux, портируются на BG/P

Компиляторы Blue Gene

- IBM XL компиляторы (xlc, xlf77, xlf90)
- Компиляция программ производится на front end узлах
  - Fortran: mpixlf, mpixlf90, mpixlf95
  - C: mpixlc
  - C++: mpixlcxx
- GNU компиляторы mpicc
- MASS математическая библиотека

OpenMP

- _r суффикс для имени компиляторов например, mpixlc_r

- –qsmp=omp

  указание компилятору интерпретировать OpenMP директивы

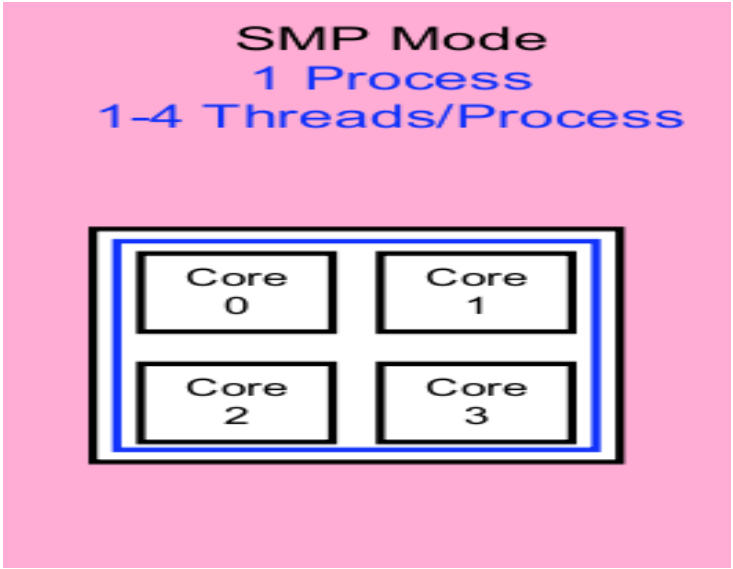
- Автоматическое распараллеливание

  -qsmp

Режимы использования ядер

- 3 режима

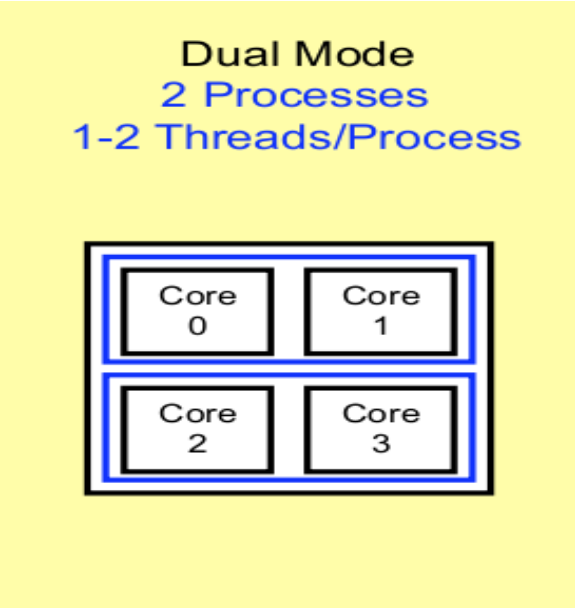
  - SMP: 1 MPI процесс из 4 SMP нитей,
    2 Гб памяти
    - mode smp



Режимы использования ядер

- 3 режима

  - DUAL: 2 MPI процесса по 2 SMP нити,
    1 Гб памяти на MPI процесс
    - mode dual

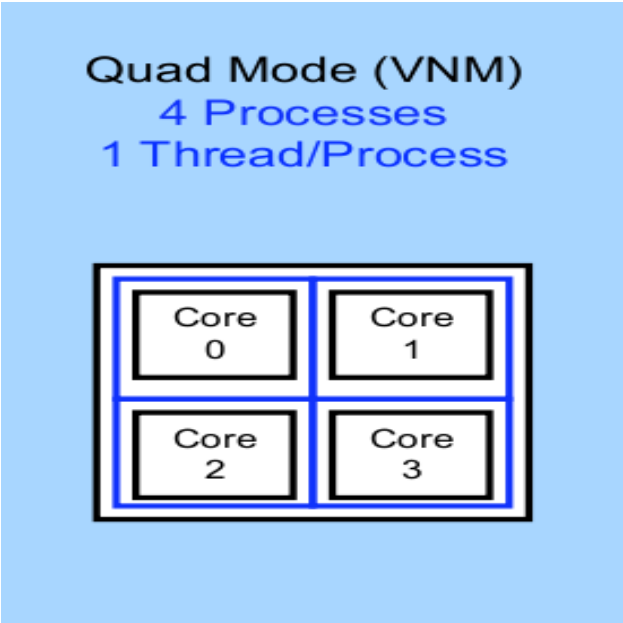


Режимы использования ядер

- 3 режима

VNM: 4 MPI процесса

- mode vn

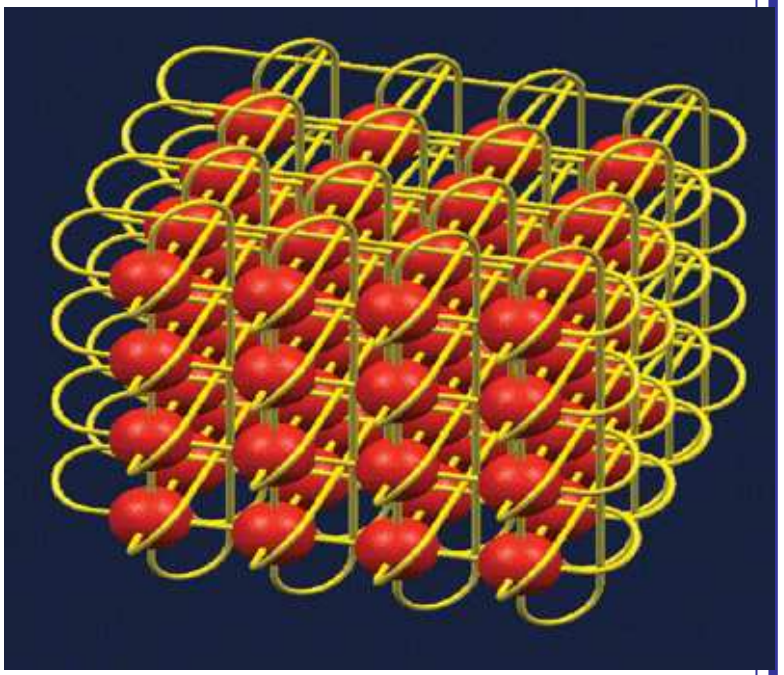


Процессорные партиции

- Подмножества вычислительных узлов, выделяемых задаче
- Каждой задаче выделяется своя партиция
- Загрузка задачи на исполнение производится независимо от других задач
- Размер партиции определяется кратным 32
- (на текущий момент на системе ВМК - кратным 128)
- Для партиций размером кратным 512 поддерживается топология тора

Назначение процессов на процессоры (mapping)

Распределение процессов
по процессорам по умолчанию:
XYZT, где <XYZ> - координаты процесса в торе,
T – номер ядра внутри процесса.
Сначала увеличивается X – координата, затем Y и Z-координаты, после этого
T- номер ядра



Mapping

2 способа назначения процессов на процессоры:

- с помощью аргумента командной строки команды

  mpirun

  –mapfile TXYZ (задаем порядок TXYZ или другие перестановки X,Y,Z,T: TYXZ, TZXY и т.д.)

Mapping

- указание map- файла в mpisubmit.bg

  –e \" MPIRUN_MAPFILE = map.txt \",

  где map.txt – имя файла.

- Синтаксис файла распределения – четыре целых числа в каждой строке задают координаты для каждого MPI-процесса (первая строка задает координаты для процесса с номером 0, вторая строка – для процесса с номером 1 и т.д.).

  0 0 0 1

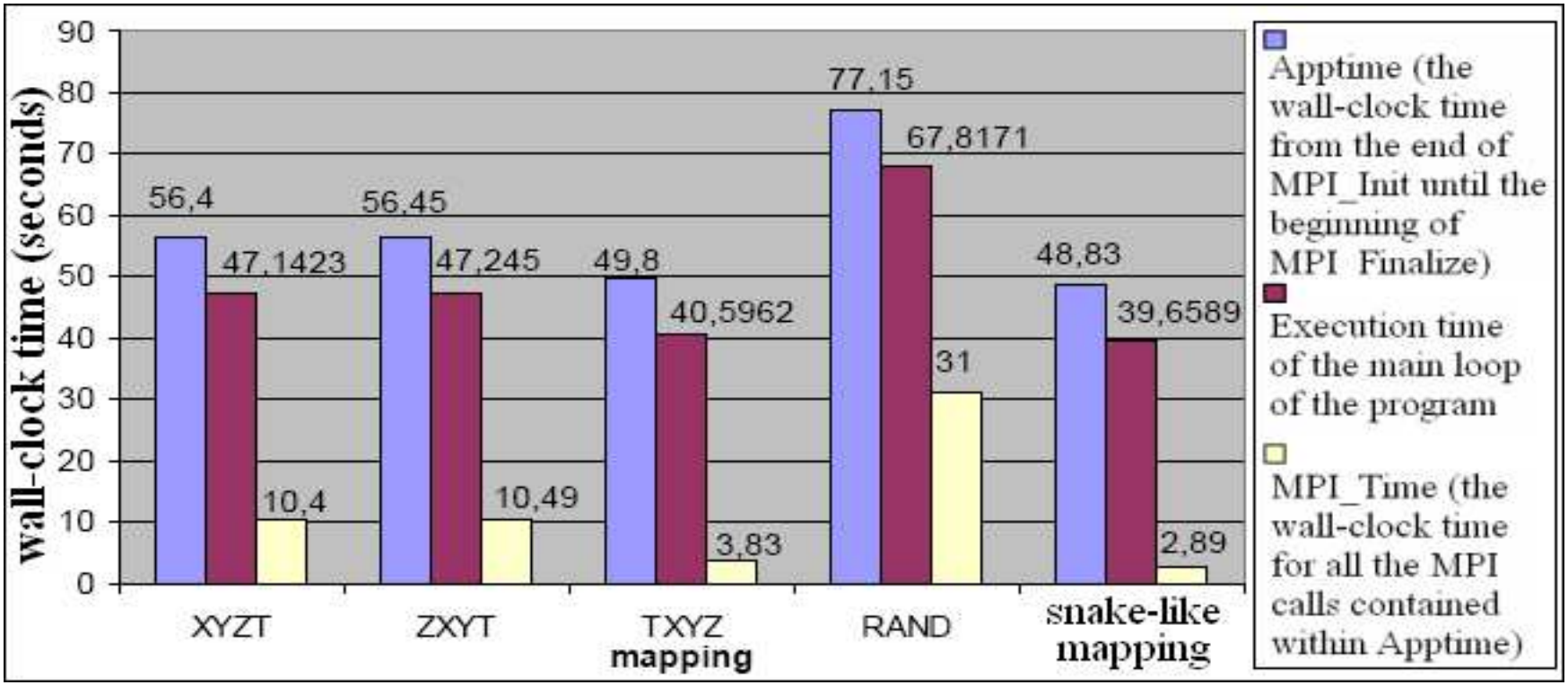
  0 0 1 1

Назначение процессов на процессоры. map-файл.

- *Очень важно, чтобы этот файл задавал корректное распределение, с однозначным соответствием между номером процесса и координатами <X, Y, Z, T>.*

Фрагмент файла, задающего mapping, сгенерированный случайным образом.

3D метод Якоби на Blue Gene/P. Распределение данных полосами. Mapping.



Среднее время выполнения **для различного mappinga** в режиме **VN** на **128 вычислительных узлах** при виртуальной топологии **1*1*512**.

Основной шаблон протокола работы пользователя (1)

1. Выход на BGP:

%ssh <опции> <логин>@bluegene1

2. Копирование файлов с локального компьютера на Blue Gene/P:

(локальная машина)

%scp example.cpp ivanov@bluegene1:~ivanov/examples

Основной шаблон протокола работы пользователя (2)

3. Компиляция MPI-программы (на языке C, C++ и Fortran90 соответственно): (BGP, front-end)

```
%mpixlc –O3 –qarch=450 –qtune=450
example.c -o c_ex
```

```
%mpixlcxx –O3 –qarch=450 –qtune=450
example.cpp -o cpp_ex
```

4. Компиляция гибридной MPI-OpenMP программы:

```
%mpixlc_r -qsmp=omp –O3 –qarch=450
–qtune=450 hw.c -o hw
```

Скрипт для запуска задач mpisubmit.bg

% mpisubmit.bg –n 128 –w 00:15:00 –e \"OMP_NUM_THREADS=4\" -m smp example – arg1 arg2

-n, --nproc	128	Запрашиваемое число узлов
-w, --wtime	00:15:00	Максимальное время выполнения
-m, --mode	smp	Режим использования ядер процессора
-e, --env		Переменные окружения
-t, --top	PREFER_TORUS	Топология
-d, --debug		Вывести содержимое командного файла на экран без постановки задачи в очередь
-h, --help		

Основной шаблон протокола работы пользователя (4)

5. Постановка MPI-программы в очередь задач с лимитом выполнения 15 минут на 32 узлах в режиме **VN** с параметром командной строки :

```
%mpisubmit.bg -w 00:15:00 -m vn -n 32
   prog – 0.1 200
```

6. Постановка MPI+OpenMP программы **prog** в очередь задач с лимитом выполнения 15 минут на 128 узлах в режиме **SMP** с 4 нитями на каждом узле, с заданием файла мэпинга map.txt и с параметром командной строки **parameter**:

```
%mpisubmit.bg -w 00:15:00 -m smp -n 128
-e \"OMP_NUM_THREADS=4  MPIRUN_MAPFILE=map.txt \"
 prog -- parameter
```

Основной шаблон протокола работы пользователя (5)

6. Информация о команде

%mpisubmit.bg –help

7. Проверка состояния очереди задач:

%llq

8. Удаление задачи из очереди:

%llcancel <task_id>

Задание 3.

- Исследование эффективности реализации 3-мерной задачи Дирихле для уравнения Лапласа на вычислительных системах Blue Gene/P и Ломоносов