Спецкурс «Суперкомпьютерные вычисления»

Эффективное параллельное программирование с учетом особенностей IBM Blue Gene/P

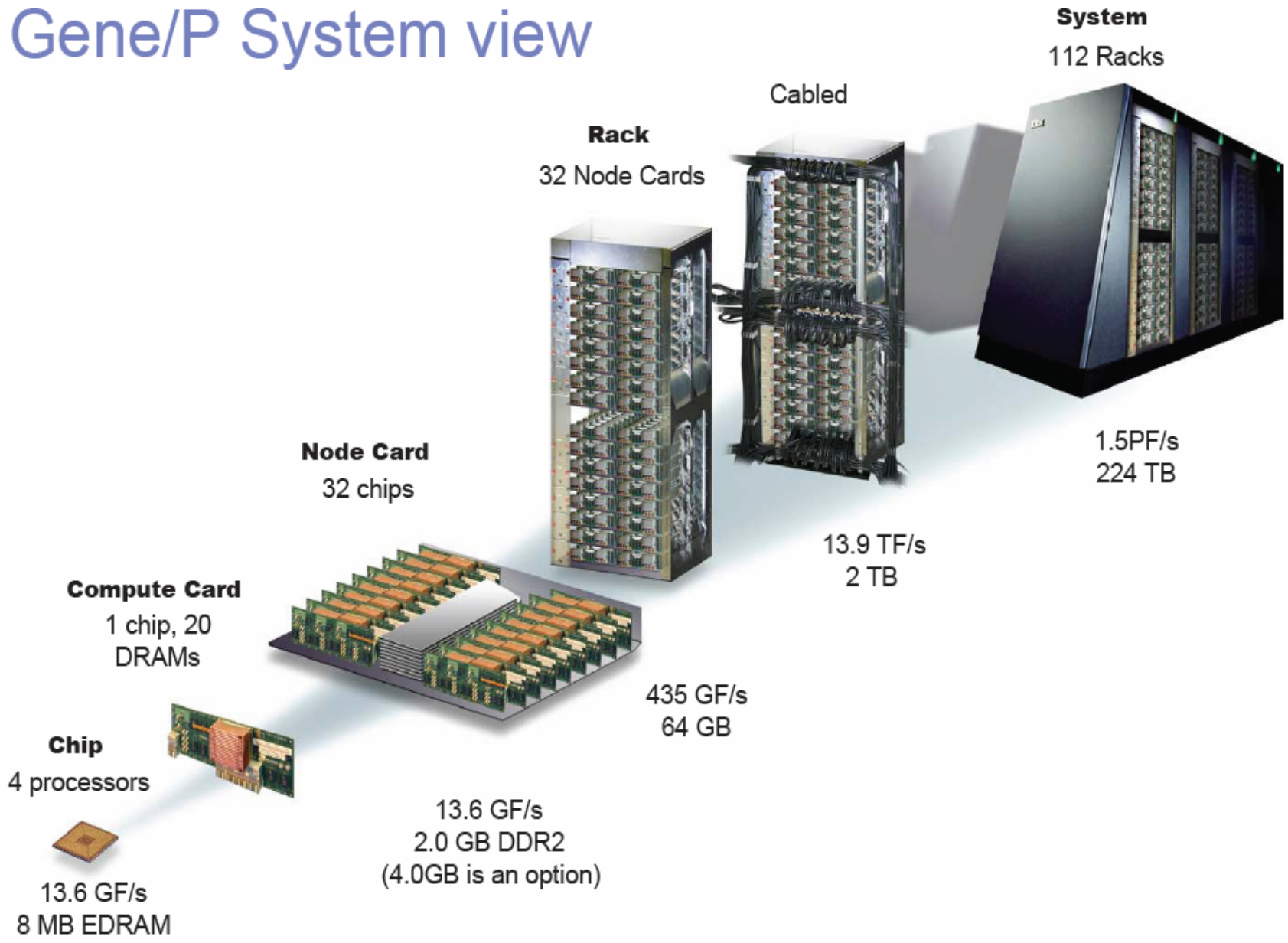
Федулова И.А.
инженер-программист, к.ф.-м.н.
IBM Russian Systems and Technology Laboratory
irina@ru.ibm.com

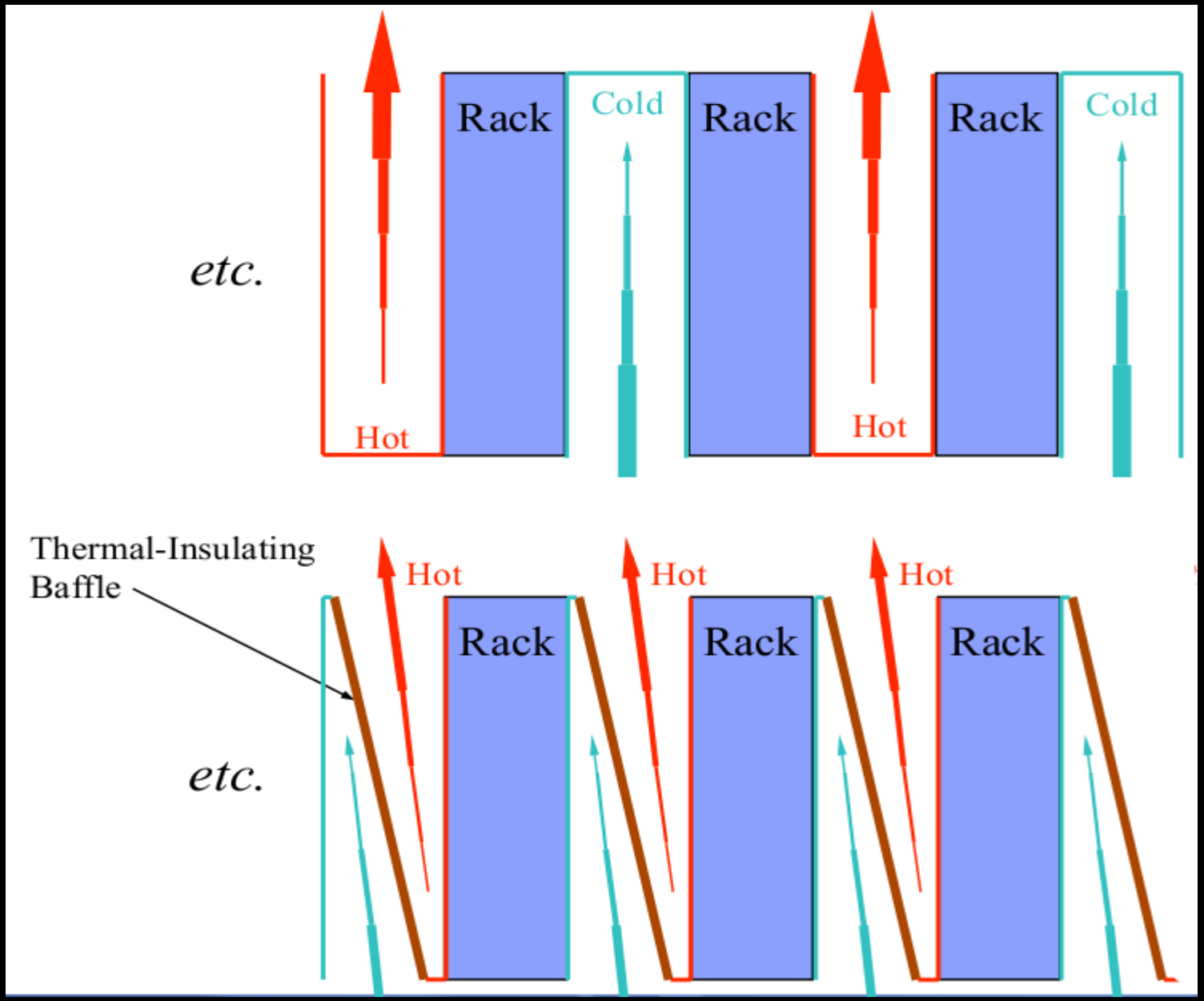
01.03.2010

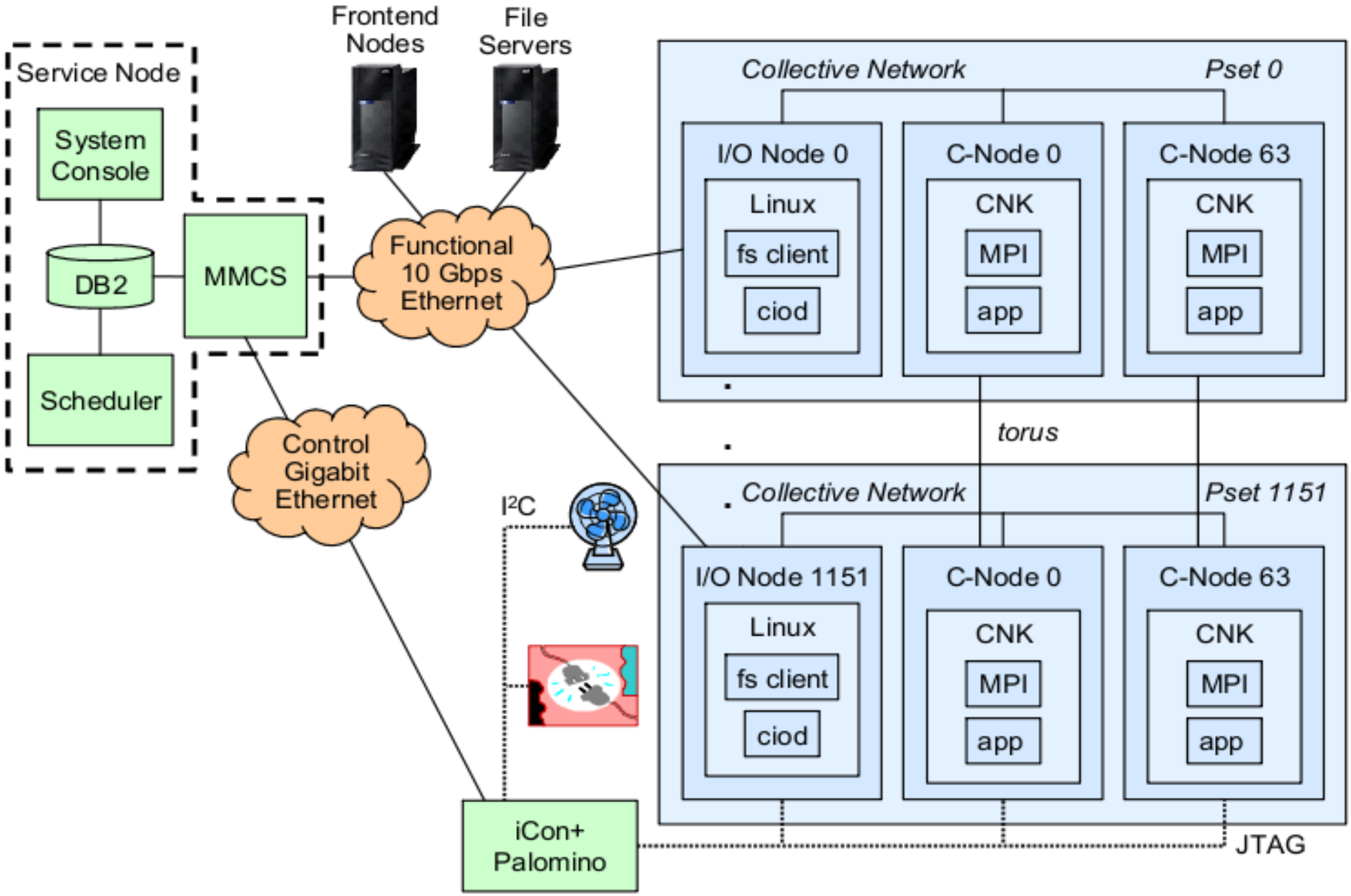
План лекции

- Особенности архитектуры
  - Режимы использования ядер процессоров
  - Коммуникационные сети
  - Размещение задач по процессорам (mapping)
- Blue Gene/P на ВМК
  - Логин
  - Файловая система
  - Компиляция программ
  - Запуск задач
  - Отладка
- Эффективное параллельное программирование
  - Общие рекомендации
  - Создание физически прямоугольных коммуникаторов
  - Коллективные операции
  - Использование библиотек
  - Альтернативы MPI
- Оптимизация
  - Измерение производительности
- Дополнительный материал *
  - Тонкая настройка MPI
  - Compiler-friendly программирование

Blue Gene/P System view

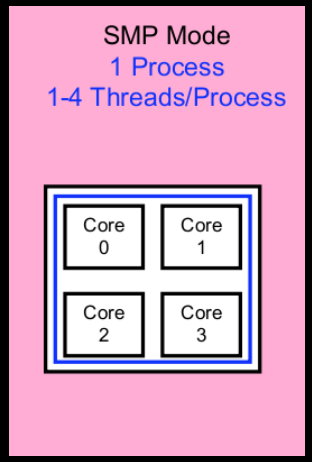


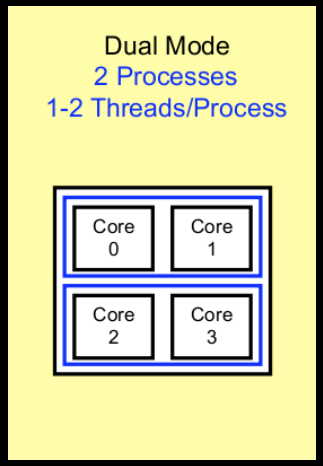

Rack
Cold
Rack
Rack
Cold
etc.
Hot
Hot
Thermal-Insulating
Baffle
Hot
Hot
Hot
Rack
Rack
Rack
etc.

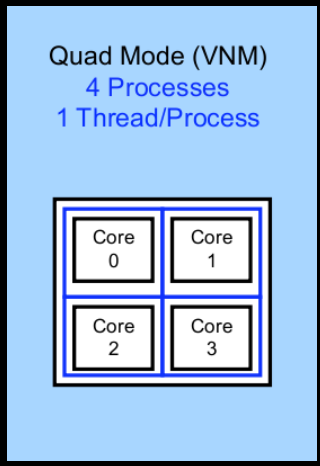

Frontend Nodes
File Servers
Service Node
System Console
DB2
MMCS
Scheduler
Functional 10 Gbps Ethernet
Control Gigabit Ethernet
I²C
iCon+ Palomino
Collective Network
Pset 0
I/O Node 0
Linux
fs client
ciod
C-Node 0
CNK
MPI
app
C-Node 63
CNK
MPI
app
torus
Collective Network
Pset 1151
I/O Node 1151
Linux
fs client
ciod
C-Node 0
CNK
MPI
app
C-Node 63
CNK
MPI
app
JTAG

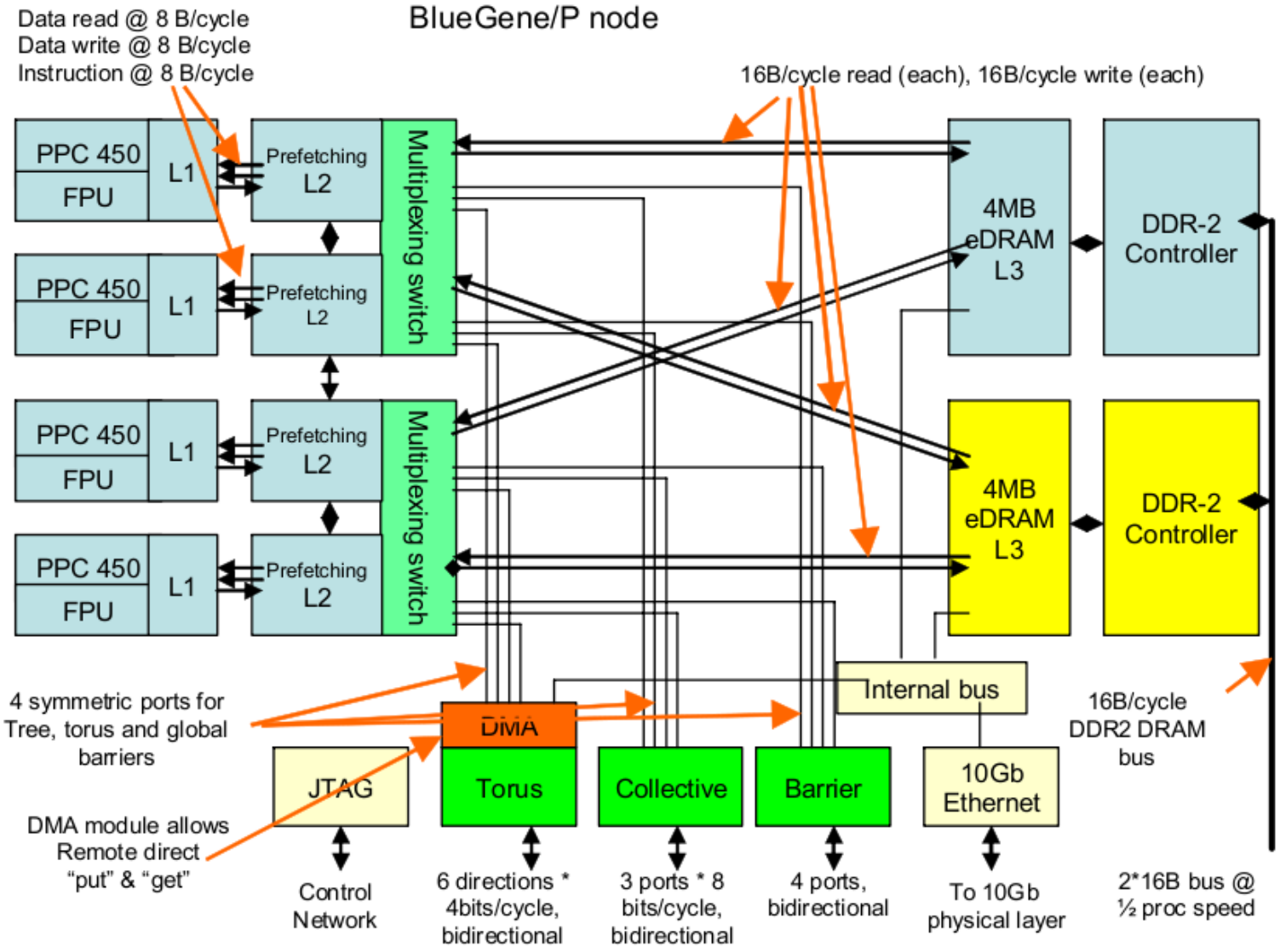
Характеристики вычислительного узла

- 4 ядерный 32-битный процессор PowerPC 850 Мгц
  - Двойное устройство для работы с числами с плавающей запятой двойной точности
  - 2 Гб памяти
  - Работает под управлением облегченной ОС
    - Создание процессов и управление ими
    - Управление памятью
    - Отладка процессов
    - Ввод-вывод
  - Объем виртуальной памяти равен объему физической
  - 3 режима использования ядер
    - SMP: 1 MPI процесс из 4 SMP нитей, 2 Гб памяти
    - DUAL: 2 MPI процесса по 2 SMP нити, 1 Гб памяти на MPI процесс
    - VNM: 4 MPI процесс





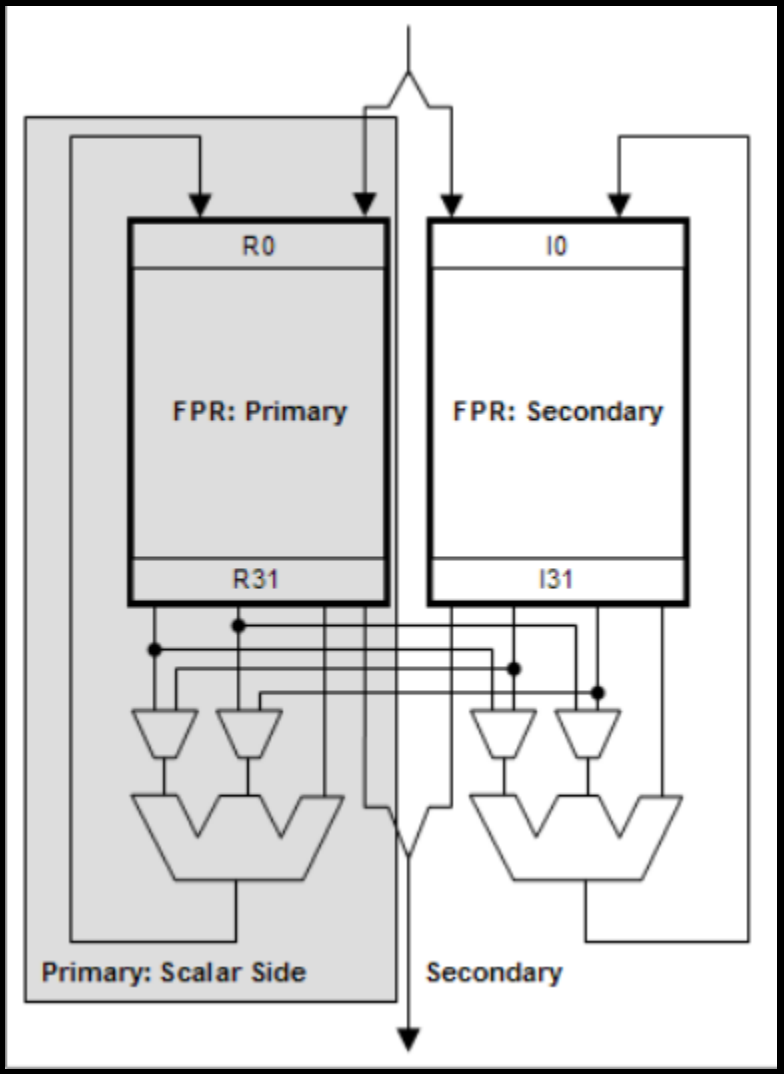



BlueGene/P node
Data read @ 8 B/cycle
Data write @ 8 B/cycle
Instruction @ 8 B/cycle
16B/cycle read (each), 16B/cycle write (each)
PPC 450
FPU
L1
Prefetching
L2
Multiplexing switch
4MB
eDRAM
L3
DDR-2
Controller
Internal bus
DMA
JTAG
Torus
Collective
Barrier
10Gb
Ethernet
4 symmetric ports for
Tree, torus and global
barriers
DMA module allows
Remote direct
"put" & "get"
16B/cycle
DDR2 DRAM
bus
Control
Network
6 directions *
4bits/cycle,
bidirectional
3 ports * 8
bits/cycle,
bidirectional
4 ports,
bidirectional
To 10Gb
physical layer
2*16B bus @
½ proc speed

Система памяти

Cache	Total per node	Size	Replacement policy	Associativity
L1 instruction	4	32 KB	Round-Robin	▸ 64-way set-associative ▸ 16 sets ▸ 32-byte line size
L1 data	4	32 KB	Round-Robin	▸ 64-way set-associative ▸ 16 sets ▸ 32-byte line size
L2 prefetch	4	14 x 256 bytes	Round-Robin	▸ Fully associative (15-way) ▸ 128-byte line size
L3	2	2 x 4 MB	Least recently used	▸ 8-way associative ▸ 2 bank interleaved ▸ 128-byte line size
Double data RAM (DDR)	2	▸ Minimum 2 x 512 MB ▸ Maximum 4 GB	N/A	▸ 128-byte line size

Double Hammer FPU

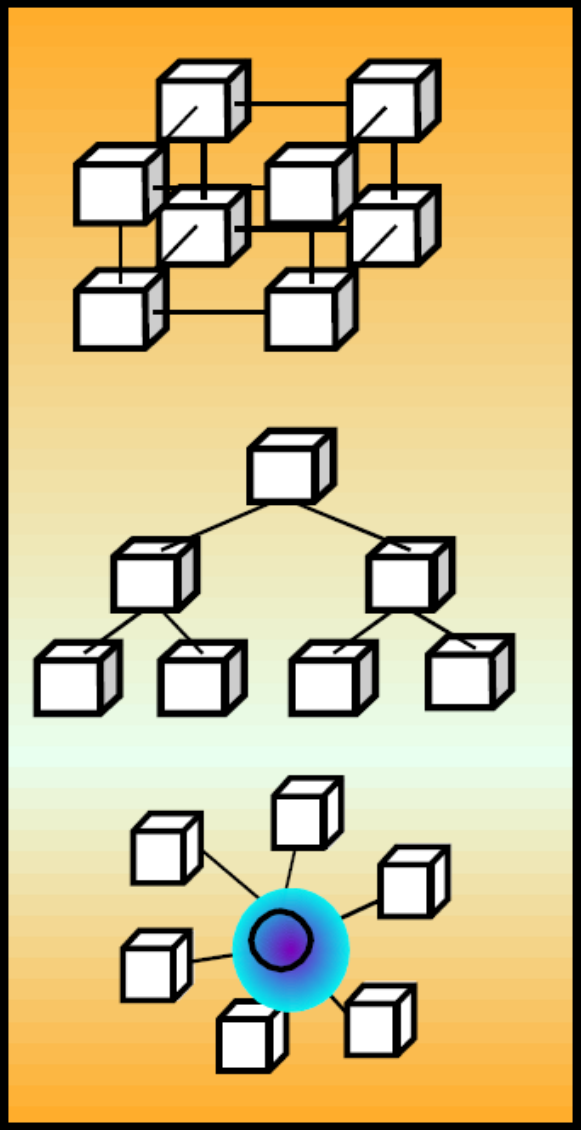


- SIMD инструкции могут выполняться одновременно на двух FPU
- Параллельные операции load/store
- Данные **должны** быть выровнены по 16-байтовой границе
  - Иначе производительность будет значительно снижена
  - Даже хуже, чем при использовании только одного FPU
- Компилятор сможет сгенерировать SIMD инструкции, только если данные в памяти расположены подряд (stride-one access)
  - Хотя при более высоких (-O4, -O5) уровнях оптимизации компилятор попытается сгенерировать SIMD инструкции и для данных, расположенных не подряд
  - -O3 –qarch=450d –qtune=450

Узлы ввода-вывода

- Используются для связи вычислительных узлов с файловой системой
- В целом точно такие же, как и вычислительные узлы, но есть отличия:
  - Установлена полноценная ОС
  - Отсутствует подключение к сети-тору
  - Имеется подключение к 10 Гб сети Ethernet

Коммуникационные сети



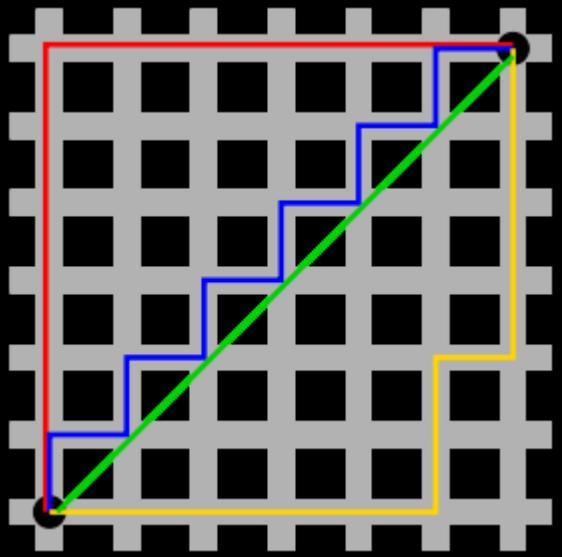
3-мерный тор
- Виртуальная аппаратная маршрутизация без буферизации
- 3.4 Гбит/с на всех 12 портах (5.1 ГБ/с на узле)
- Аппаратные задержки: 0.5 мс между соседними узлами, 5 мс между самыми далекими
- Задержки MPI: 3мс между соседними узлами, 10 мс между самыми далекими
- Основная коммуникационная сеть
- Используется в том числе для многих коллективных операций

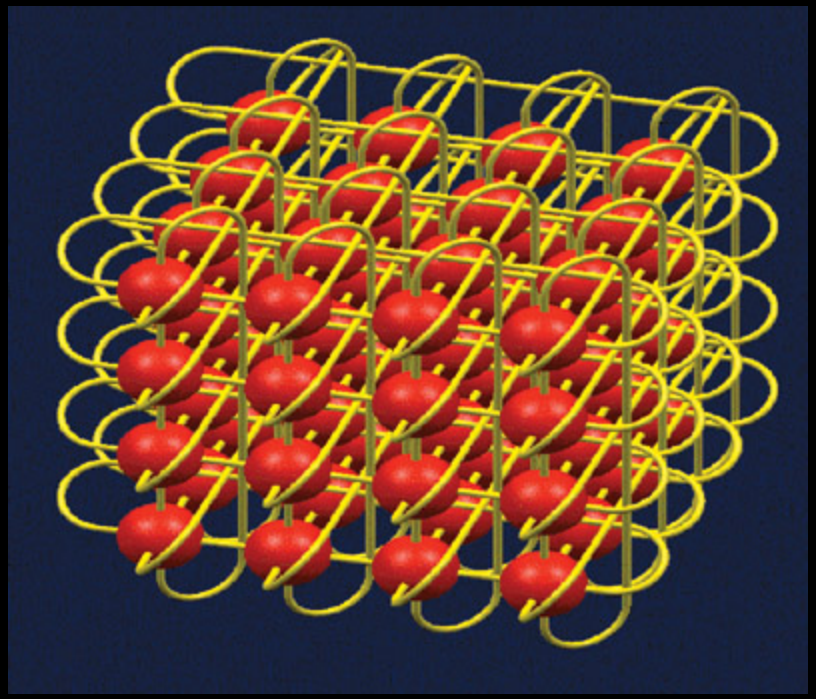
Коллективная сеть – дерево
- Для глобальной коммуникации один-ко-всем (broadcast, reduction)
- 6.8 ГБ/с на порт
- Задержка на один проход дерева: аппаратная 1.3 мс, MPI 5 мс
- Соединяет все вычислительные узлы и узлы ввода-вывода
- Используется для коллективных операций и коммуникатора MPI_COMM_WORLD

Высокоскоростная сеть для глобальных прерываний
- Задержка на оповещение всех узлов: аппаратная 0.65 мс, MPI 5 мс
- Для MPI_Barrier

Размещение процессов MPI по узлам (mapping)

- Латентность зависит от расстояния, на которое передаются сообщения
- Hop distance — количество промежуточных узлов
  - Расстояние Манхеттена





Размещение процессов MPI по узлам (mapping)

- Доступные размеры физических решеток
  - `32:   4 x 4 x 2`
  - `64:   4 x 4 x 4`
  - `128:  4 x 4 x 8`
  - `256:  8 x 4 x 8`
  - `512:  8 x 8 x 8`
  - `1024: 8 x 8 x 16`
  - `1024: 8 x 16 x 8`
  - `2048: 8 x 16 x 16`

- Чтобы указать конкретную размерность решетки для 1024 узлов, в командном файле необходимо указать параметры

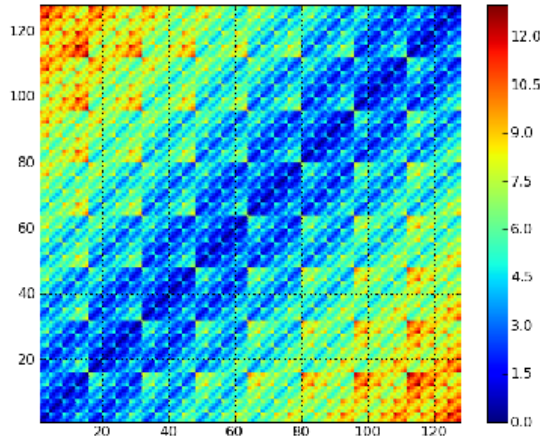
```
bg_rotate=false
bg_shape=8x8x16 либо bg_shape=8x16x8
```

Матрица расстояний между узлами

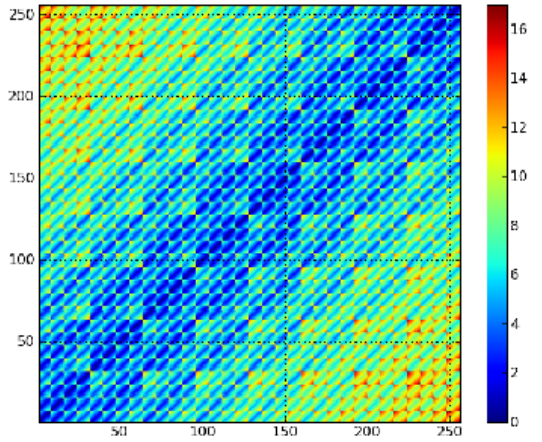
- Зависит от
  - размера раздела
  - способа нумерации узлов
- Не зависит от программы

- Как получить матрицу расстояний?
  - Написать простую программу, которая ее рассчитывает

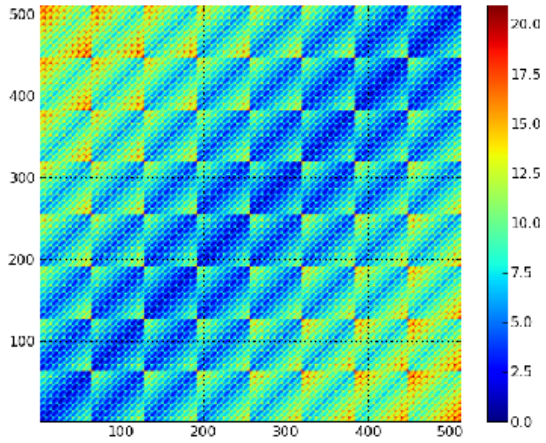
Примеры матриц расстояния между узлами



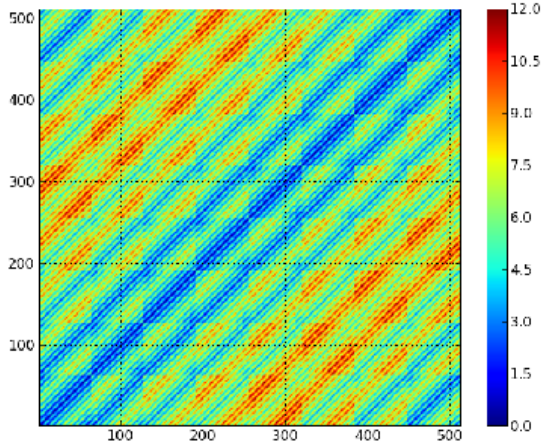
$Np = 128$



$Np = 256$



$Np = 512$ (MESH)



$Np = 512$ (TORUS)

Матрица коммуникации MPI программы

- Зависит от программы и ее входных параметров
- Не зависит от того, на каком (супер)компьютере программа запущена

- Как получить матрицу коммуникации?
  - Использовать HPC Toolkit

    Указать при линковке

    `-lmpitrace -L<путь к библиотеке>`

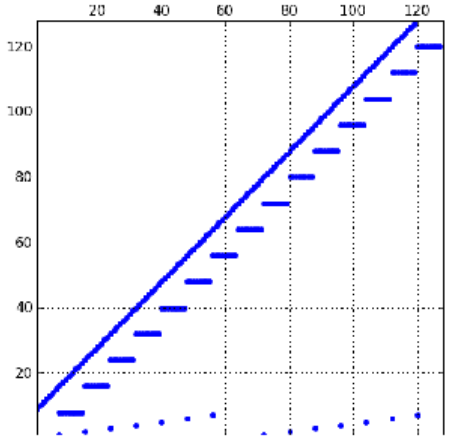
    При запуске указать переменные окружения

    `-e TRACE_SEND_PATTERN=yes -e TRACE_ALL_EVENTS=yes`

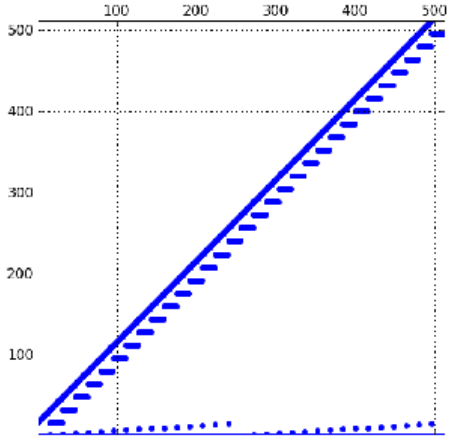
    Запустить программу

    Получить файл `send_bytes.matrix`
    - Содержит квадратную матрицу `Nproc x Nproc` чисел типа `double`
    - `(i,j)` элемент матрицы = количество байт, переданных от процессора i процессору j
  - Самостоятельно инструментировать MPI_Send/MPI_Recv

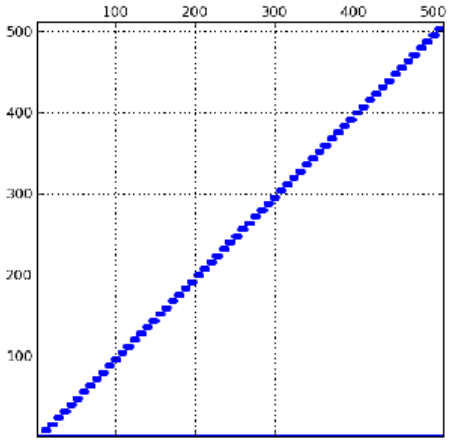
Примеры матриц коммуникации



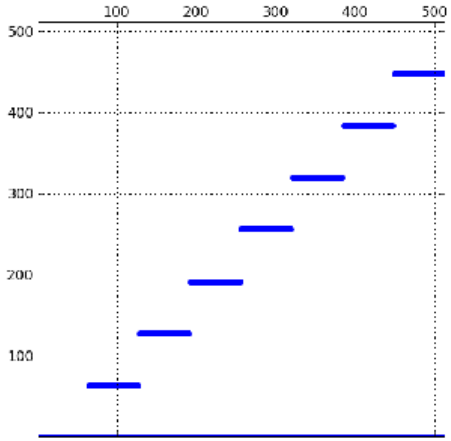
PBLAS: 128 nodes, SMP



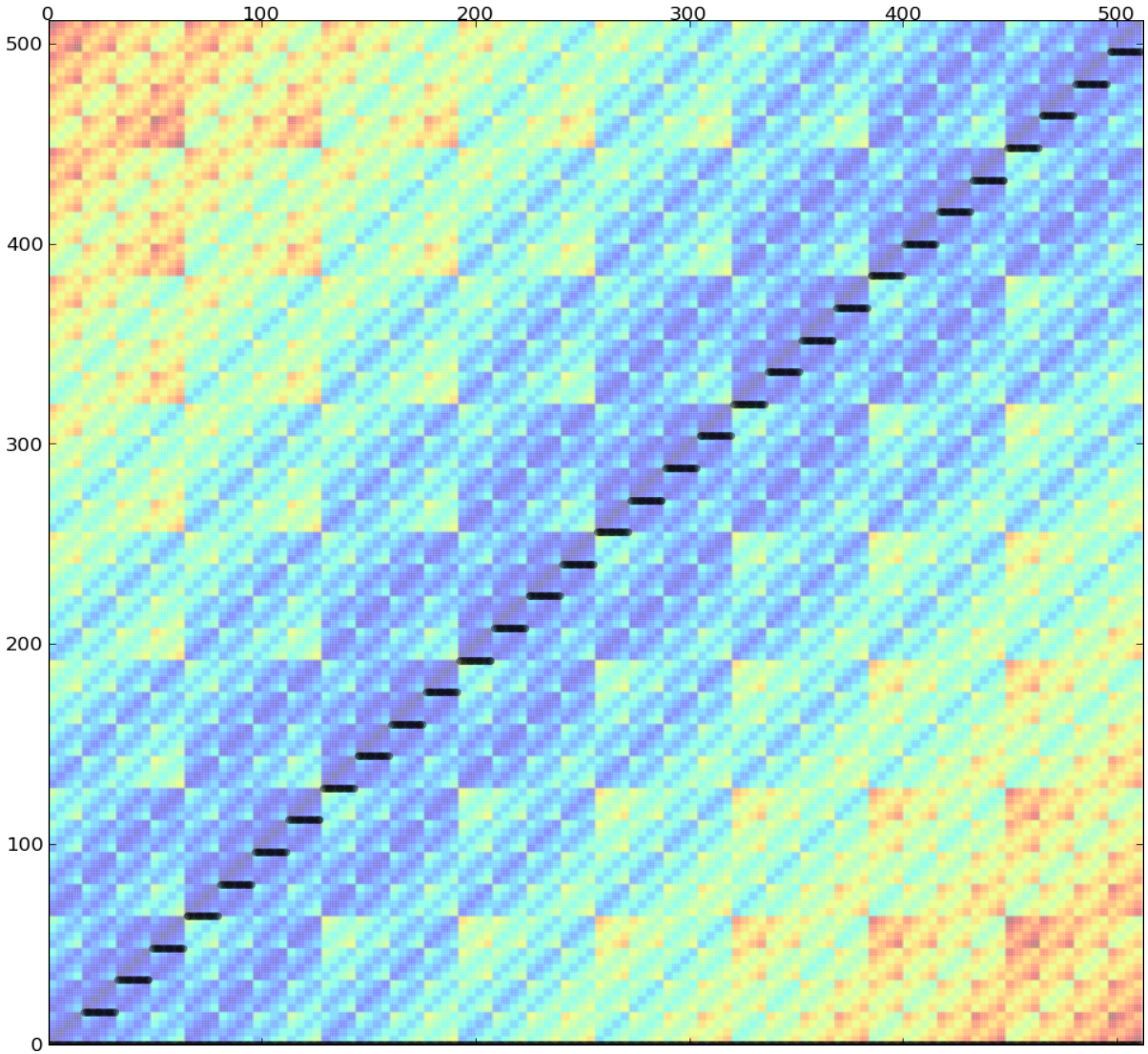
PBLAS: 128 nodes, DUAL

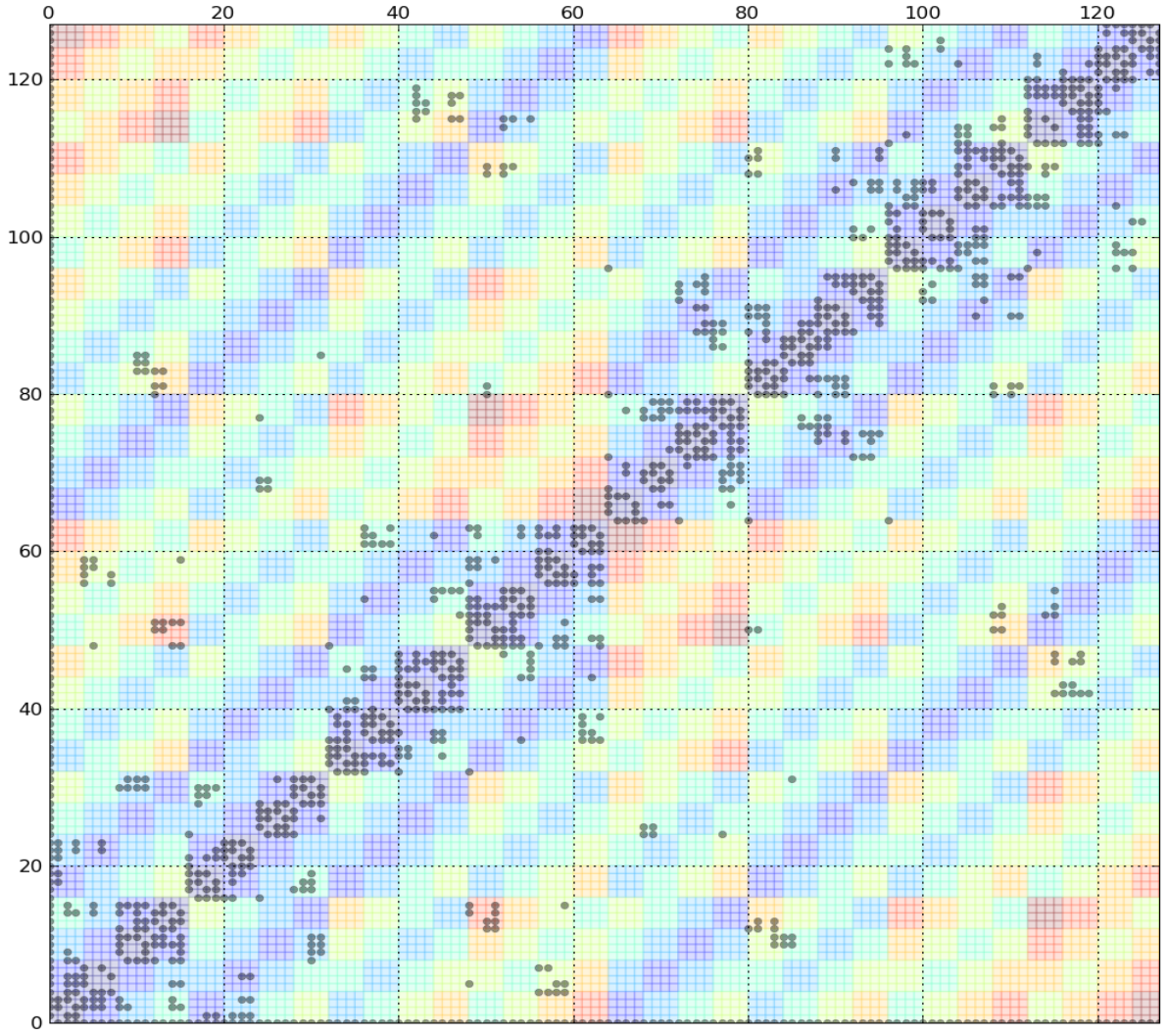


FAT_SGEMM, *blk* = 8, 128 nodes, VN



FAT_SGEMM, *blk* = 64, 128 nodes, VN


0
100
200
300
400
500
500
400
300
200
100
0


0
20
40
60
80
100
120
120
100
80
60
40
20
0

Размещение процессов MPI по узлам (mapping)

- Выводы
  - Необходимо учитывать то, как будет отображаться коммуникационная матрица на топологию на этапе написания программы
  - Можно экспериментировать со стандартными мэппингами
  - Можно создавать mapping-файлы вручную

IBM Blue Gene/P на ВМК

- 2 стойки по 1024 4-ядерных процессора
  - Общий объем ОЗУ: 4Тб
  - Пиковая производительность: 27.2 Тфлоп/с
  - Реальная производительность по тесту Linpack: 23.2 Тфлоп/с
    - 85% от пиковой

IBM Blue Gene/P на ВМК

- Доступ к системе ограничен
  - Внутри факультета ВМК с любого IP адреса
  - Извне — только с отдельных IP адресов (индивидуально для пользователей)
  - При первом логине необходимо сменить пароль

```
$> ssh username@bluegene1.cs.msu.su
$> passwd
```

- Файловая система
  - /home/username
    - постоянное хранение файлов
    - редактирование текстов программ
    - компиляция
    - обработка результатов и визуализация
  - /gpfs/data/username
    - временное хранение (только на время запуска и выполнения программы)
    - перез запуском программы необходимо переписать исполняемый файл и входные файлы на /gpfs
    - после окончания счета необходимо переместить результаты в /home

Компиляция программ

Компилятор		C/C++		Fortran			
		C	C++	Fortran 77	Fortran 90	Fortran 95	Fortran 2003
GCC		`mpicc`	`mpicxx`	`mpif77`	–	–	–
IBM XL		`mpixlc`	`mpixlcxx`	`mpixlf77`	`mpixlf90`	`mpixlf95`	`mpixlf2003`
	потоко-безопасная версия	`mpixlc_r`	`mpixlcxx_r`	`mpixlf77_r`	`mpixlf90_r`	`mpixlf95_r`	`mpixlf2003_r`

Компиляция программ

```
irina@fen1:~/gpfs/helloworld> cat hello.cpp
#include <mpi.h>
#include <iostream>

int main (int argc, char *argv[])
{
    MPI_Init (&argc, &argv);

    int sz, id;
    MPI_Comm_size (MPI_COMM_WORLD, &sz);
    MPI_Comm_rank (MPI_COMM_WORLD, &id);

    std::cout << "hi! i'm " << id <<  " of " << sz << std::endl;

    MPI_Finalize();
}

irina@fen1:~/gpfs/helloworld> mpicxx hello.cpp -o hello
irina@fen1:~/gpfs/helloworld> ls -l hello
-rwxr-xr-x 1 irina cmc 14550266 2009-11-02 12:07 hello
```

Ключи компилятора

- –qarch=450 –qtune=450
  - Инструкции только для одного из двух FPU
  - Используйте, если данные не выровнены по 16-байтовой границе
- –qarch=450d –qtune=450
  - Выравнивание!
- –O3 (–qstrict)
  - Минимальный уровень оптимизации под Double Hammer FPU
- –O3 –qhot (= –qsimd)
- –O4 (–qnoipa)
- –O5
- –qdebug=diagnostic
  - Информация по SIMD-инструкциям (только с qhot)
- –qreport –qlist –qsource
  - Много полезной информации в .lst файле

Постановка заданий в очередь

- Система управления заданиями IBM LoadLeveler
  - Оптимизирует использование имеющихся вычислительных ресурсов
    - Учет приоритета задач и пользователей
    - Динамическое распределение ресурсов
  - Очередь заданий

- Основные команды для системы LoadLeveler
  - llsubmit — постановка задачи в очередь

```
irina@fen1:~/gpfs/helloworld> llsubmit test.jcf
llsubmit: The job "fen1.bg.cmc.msu.ru.42037" has been submitted.
```

  - llq — просмотр текущего статуса очереди

```
irina@fen1:~/gpfs/helloworld> llq
Id                       Owner      Submitted   ST PRI Class        Running On
------------------------ ---------- ----------- -- --- ------------ -----------
fen1.42037.0             irina      11/1  20:56 R  50  n128_m15     fen1
fen1.42024.0             kovalev    11/1  21:21 R  50  n512_h12     fen1
fen1.42016.0             ruby_rush  11/1  13:45 I  50  n2048_m03
fen1.42025.0             psmart     11/1  21:21 NQ 50  n1024_h24
```

Удаление задач из очереди

– llcancel — удаление задач из очереди, если задача еще не запущена на счет (статус I, NQ, P)

```
irina@fen1:~/gpfs/helloworld> llcancel 42037
llcancel: Cancel command has been sent to the central manager.
```

– **Важное замечание: у Blue Gene/P на ВМК есть особенность**
  – если задача уже запущена (статус R),
  … лучше ее **не удалять**, иначе может возникнуть проблема с запуском последующих задач
  – есть надежда, что эта проблема скоро будет решена администраторами

Скрипт mpisubmit.bg

- Составляет командный файл для LoadLeveler-а
- Ставит задачу в очередь

```
irina@fen1:/gpfs/data/irina/helloworld> mpisubmit.bg -n 128 -w 00:15:00 \
                                                     -e OMP_NUM_THREADS=4 \
                                                     -m smp hello -- 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`		

Командный файл изнутри

- Структура файла
  - комментарии начинаются с #
  - команды LoadLeveler-а начинаются с #@
- Файл содержит
  - название исполняемого файла
  - класс задачи (наименование очереди)
  - требуемые ресурсы (число узлов, время выполнения)
  - имена входных и выходных файлов

```
#@ job_type = bluegene
#@ bg_size = 128                        # количество запрашиваемых узлов
#@ wall_clock_limit = 00:15:00          # требуемое время
#@ class = n128_m15                     # класс задачи
#@ bg_connection = PREFER_TORUS
#@ output = hello.$(jobid).out          # файл стандартного вывода
#@ error = hello.$(jobid).err           # файл потока ошибок
#@ notification = never
#@ queue
/bgsys/drivers/ppcfloor/bin/mpirun \
      -env CMC_SITE_SPECIFIC=n128_m15 \
      -np 128 \
      -mode smp \                       # режим использования ядер на процессоре
      -exe hello \    # относительный путь, но тогда llsubmit из каталога на gpfs
      -verbose 2        ... либо абсолютный путь
```

Параметр mapping

- Стандартные мэппинги можно задать параметром для скрипта mpisubmit.bg
  - **-e BG_MAPPING=TXYZ**
- Любой мэппинг можно указать в командном файле как параметр mpirun
  - **-mapfile /gpfs/data/irina/hello/mapfile.txt**

```
#@ job_type = bluegene
#@ class = large
#@ wall_clock_limit = 02:00:00
#@ error = cpmd.$(jobid).err
#@ output = cpmd.$(jobid).out
#@ bg_size = 512
#@ bg_connection = PREFER_TORUS
#@ queue
/bgsys/drivers/ppcfloor/bin/mpirun \
        -exe /gpfs/data/irina/bin/cpmd.smp+mpi.x \
        -args "1neutral.inp" \
        -env "PP_LIBRARY_PATH=/gpfs/data/irina/cpmd/pp_library/ BG_MAPPING=TXYZ OMP_NUM_THREADS=4" \
        -mapfile randomMapping_n512.txt \
        -mode smp -verbose 2
```

```
5 2 6 0
1 5 0 0
6 7 4 0
2 3 7 0
1 2 0 0
4 3 3 0
7 7 7 0
0 2 1 0
5 4 3 0
4 2 1 0
5 6 3 0
5 5 2 0
4 7 7 0
```

Отладка

- На Blue Gene нет виртуальной памяти, поэтому необходимо тщательно следить за утечками памяти
- Ключ компилятора -g
- addr2line
  - Утилита для анализа core файлов
- Application killed with signal 6
  - Недостаточно памяти
- Application killed with signal 7 (memory alignment error)
  - BG_MAXALIGNEXP=1
    - процесс умирает при первой ошибке выравнивания (полезно при оптимизации)
  - BG_MAXALIGNEXP=-1
    - игнорировать ошибки выравнивания

Общие рекомендации по написанию MPI программ

- Стремитесь выполнять коммуникации и вычисления одновременно
- Избегайте несбалансированной нагрузки
- Избегайте буферизации при отправке сообщения
  - Recv заранее
- Избегайте использования специальных типов данных
  - перед отправкой система их будет упаковывать, тратя на это время
- Специфичные для Blue Gene рекомендации:
  - Используйте прямоугольные коммуникаторы
  - Используйте коллективные операции вместо большого числа операций точка-точка
  - Следите за выравниванием данных на 16-байтовую границу

Прямоугольный коммуникатор

```
#include <mpix.h>
int MPIX_Cart_comm_create(MPI_Comm *cart_comm);
```

- Создает четырехмерный декартов коммуникатор, логические размерности которого совпадают с физическими размерами аппаратного раздела (partition)
- Координаты в логическом коммуникаторе и физическом partition совпадают
- Коллективная операция

Характеристики коллективных операций

- Для Blue Gene была разработана специальная реализация MPICH2
- Оптимизированы коллективные операции
- Производительность коллективных операций зависит от
  - Коммуникатора
  - Типа данных
- Таблица полностью есть в Redbook-е

MPI routine	Communicator	Data type	Network	Latency	Bandwidth
MPI_Barrier	MPI_COMM_WORLD	N/A	Global Interrupts	1.25 µs	N/A
	Rectangular	N/A	Torus	10.96 µs	N/A
	All other subcommunicators	N/A	Torus	22.61 µs	N/A
MPI_Bcast	MPI_COMM_WORLD	Byte	Collective for latency, torus for BW	3.61µs	2047 MBps[a]
	Rectangular	Byte	Torus	11.58 µs	2047 MBps[a]
	All other subcommunicators	Byte	Torus	15.36 µs	357 MBps
MPI_Allreduce	MPI_COMM_WORLD	Integer	Collective for latency, torus for BW	3.76 µs	780 MBps[a]
		Double	Collective for latency, torus for BW	5.51 µs	363 MBps[a]
	Rectangular	Integer	Torus	17.66 µs	261 MBps[a]
		Double	Torus	17.54 µs	363 MBps[a]
	All other subcommunicators	Integer	Torus	38.06 µs	46 MBps

Использование оптимизированных библиотек

- ESSL
  - линейная алгебра
  - матричные операции
  - задачи на собственные значения
  - преобразование Фурье
  - сортировка и поиск
  - интерполяция
  - генерация случайных чисел

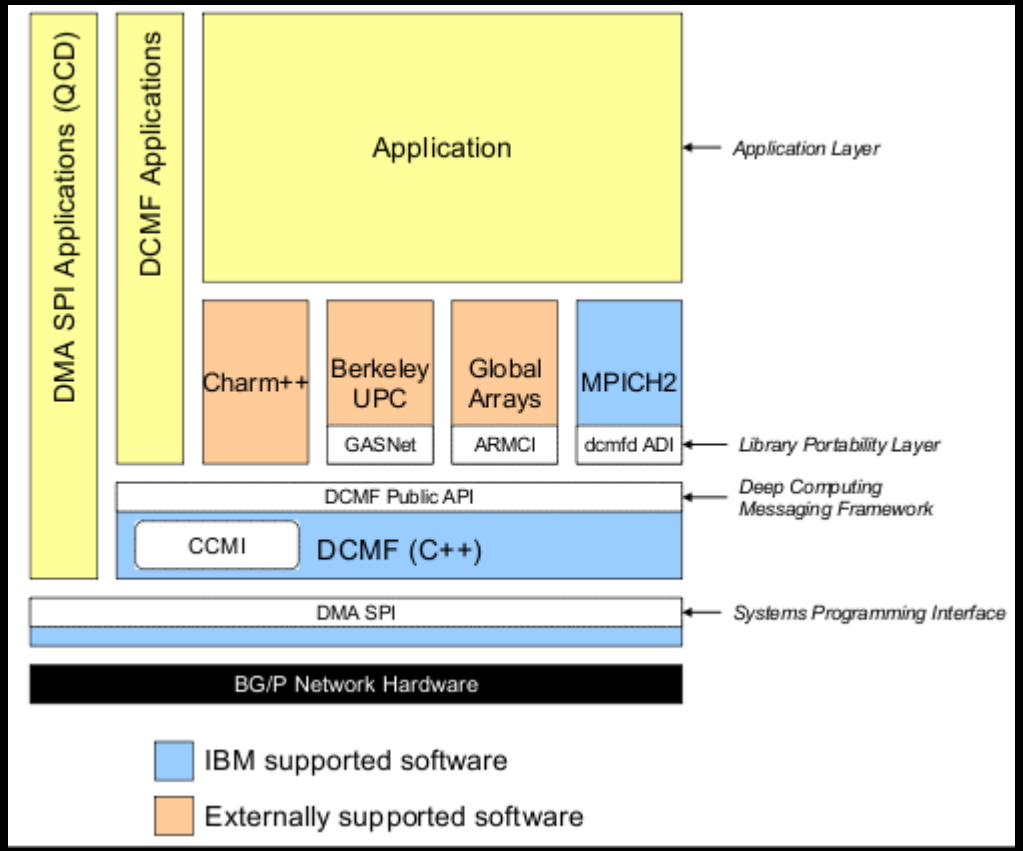
  - не параллельная! Использует OpenMP, может работать только в рамках одного узла
  - Используйте ScaLAPACK вместо Parallel ESSL

```
-lesslbg -L/bgsys/ibm_essl/sles10/prod/opt/ibmmath/lib
-lesslsmpbg
```

- MASS
  - sin, cos, tan, sqrt, rsqrt, exp, log, …
  - IBM XL компилятор линкует MASS автоматически

Альтернативы MPI

- DCMF
- Надстройки над DCMF
  - MPICH2
  - Global Arrays
  - Charm++
  - Berkeley UPC
  - ARMCI: Aggregate Remote Memory Copy Interface



Измерение производительности

- Производительность на уровне MPI
  - HPC Toolkit: libmpitrace
  - TAU: `http://www.cs.uoregon.edu/research/tau/`
  - mpiP: `http://mpip.sourceforge.net/`
  - Jumpshot: `http://www.mcs.anl.gov/research/projects/perfvis/software/viewers/`
  - …
- Производительность на уровне процессора
  - HPC Toolkit: Xprofiler, libhpm
  - PAPI: `http://www.redbooks.ibm.com/abstracts/redp4256.html`
  - UPC wrappers: `http://hpc.cs.msu.su/en/bgp/development/performance`
  - gprof

Измерение производительности вручную

http://www.fz-juelich.de/jsc/jugene/usage/FAQ/measurement

```c
#include <sys/resource.h>
#include <common/bgp_personality.h>
#include <common/bgp_personality_inlines.h>
#include <spi/kernel_interface.h>

static _BGP_Personality_t mybgp;

/* returns memory per core in MBytes */
unsigned bg_coreMB() {
  unsigned procMB, coreMB;
  Kernel_GetPersonality(&mybgp, sizeof(_BGP_Personality_t));
  procMB = BGP_Personality_DDRSizeMB(&mybgp);
  coreMB = procMB/Kernel_ProcessCount();
  return coreMB;
}

/* return maximum memory usage of process in kBytes */
unsigned bg_usedKB() {
  struct rusage usage;
  if (getrusage(RUSAGE_SELF, &usage) != 0) return 0;
  return usage.ru_maxrss;
}
```

```c
#include <common/bgp_personality.h>
#include <common/bgp_personality_inlines.h>
#include <spi/kernel_interface.h>
static _BGP_Personality_t mybgp;
static double clockspeed=0.0;

/* initialize the clockspeed scaling factor */
unsigned bg_wtime_init() {
  unsigned clockMHz;
  Kernel_GetPersonality(&mybgp, sizeof(_BGP_Personality_t));
  clockMHz = BGP_Personality_clockMHz(&mybgp);
  clockspeed = 1.0e-6/(double)clockMHz;
  return clockMHz;
}

/* return time in seconds */
double bg_wtime() {
  return ( _bgp_GetTimeBase() * clockspeed );
}
```

Источники информации

- Сайт «Высокопроизводительные вычисления на ВМК МГУ»
  http://hpc.cs.msu.su/

- **Redbook** IBM System Blue Gene Solution: Blue Gene/P Application Development
  http://www.redbooks.ibm.com/abstracts/sg247287.html

- Материалы прошлогоднего спецкурса «Терафлопные вычисления»
  http://angel.cmc.msu.ru/~ifed/teraflops/

- Сайт Blue Gene/P в суперкомпьютерном центре Юлиха (Германия)
  http://www.fz-juelich.de/jsc/jugene/usage/

- Тонкая настройка ключей компилятора
  http://www.nccs.gov/wp-content/training/2008_bluegene/ORNLCompOptimization.pdf

- Сайт центра научных вычислений бостонского университета
  http://scv.bu.edu/computation/bluegene/

Дополнительный материал *

Тонкая настройка MPI

- Все операции точка-точка и коллективные операции на субкоммуникаторах выполняются через тор

- Детерминированная маршрутизация
  - Может привести к образованию «горячих точек»
- Адаптивная маршрутизация
  - Дополнительные накладные расходы

- Маршрутизация зависит от размера сообщения
  - short protocol: 1 пакет < 244 байт: детерминированная
  - eager protocol: сообщения «средних» размеров (по умолчанию < 1200 байт): детерминированная
  - rendezvous protocol: после предварительного установления контакта используется DMA (Direct Memory Access) с адаптивной маршрутизацией
    - Максимальная эффективность по bandwidth, но и максимальная latency
  - переменная окружения DCMF_EAGER регулирует порог между eager и rendezvouz
    - Уменьшайте DCMF_EAGER, если используете много коротких сообщений и далекие расстояния
    - Увеличивайте DCMF_EAGER, если коммуникации в основном между соседями, либо если используете довольно большие сообщения

Тонкая настройка MPI

- Настройка MPI с использованием переменных окружения
  - mpisubmit –n 128 –env “DCMF_EAGER=2000” myprog

- Тонкая настройка MPI во время выполнения программы
  - В последнем драйвере Blue Gene V1R3M0 появилась возможность менять значения параметров настройки MPI программно

```
int MPIX_Get_properties(MPI_Comm comm, int *prop_array);
int MPIX_Get_property(MPI_Comm comm, int prop, int *result);
int MPIX_Set_property(MPI_Comm comm, int prop, int value);
```

```
int main(int argc, char **argv)
{
   …
   MPIX_Get_property(comm, DCMF_USE_TORUS_ALLTOALL, &result);
   if (result == 0)
      /* this causes the following MPI_Alltoall to use torus protocol */
      MPIX_Set_property(comm, DCMF_USE_TORUS_ALLTOALL, 1);

   MPIX_Get_property(comm, DCMF_USE_MPICH_ALLTOALL, &result);
   if (result == 1)
      /* turn off the mpich protocol */
      MPIX_Set_property(comm, DCMF_USE_MPICH_ALLTOALL, 0);

   MPI_Alltoall(…);
   /* this resets the MP_Alltoall algorithm selection to its previous state */
   MPIX_Set_property(comm, DCMF_USE_TORUS_ALLTOALL, 0);
   …
   …
}
```

Тонкая настройка MPI

- Параметры MPI, с которыми можно экспериментировать
  - DCMF_SENDER_SIDE_MATCHING {[0], 1}
  - DCMF_COLLECTIVE {0, [1], NOTREE}
  - Параметры, указывающие конкретный алгоритм для коллективных операций
    - DCMF_BCAST, DCMF_ALLTOALL, DCMF_ALLGATHER, DCMF_ALLREDUCE, DCMF_BARRIER, … {MPICH, TREE, BINOM, DPUT, RECT, …}
    - Система выбирает конкретный алгоритм в зависимости от коммуникатора
  - Полный список в Redbook

- STAR-MPI: Self-Tuned Adaptive Routines for MPI
  - Динамически выбирает оптимальный алгоритм для коллективных операций
  - Не нужно пересобирать программу, достаточно запустить с переменными окружения
    - DCMF_STAR=1 DCMF_STAR_VERBOSE=1
  - DCMF_STAR_THRESHOLD [=2048]: размер сообщения, начиная с которого будут использоваться функции STAR-MPI
  - DCMF_STAR_NUM_INVOCS=[10]: число вызовов коллективной операции, которое STAR-MPI использует для сбора статистики

Тонкая настройка MPI

- FastMPI
  - Используйте компиляторы из `/bgsys/drivers/ppcfloor/comm/fast/bin`
  - Только для отлаженных кодов
  - Коллективные операции не оптимизированы – ?

```
irina@fen1:~> ls /bgsys/drivers/ppcfloor/comm/fast/bin/mpi
mpicc           mpicxx          mpif90          mpixlc_r        mpixlcxx_r
  mpixlf2003_r    mpixlf77_r      mpixlf90_r      mpixlf95_r
mpich2version   mpif77          mpixlc          mpixlcxx        mpixlf2003
  mpixlf77        mpixlf90        mpixlf95
```

Compiler-friendly программирование

- Располагайте данные в непрерывных областях памяти
- Проверяйте выравнивание на 16-байтовую границу
  - __alignx не выравнивает и не проверяет на выравнивание, а сообщает компилятору, что массивы выровнены
- Группируйте операции с плавающей запятой
- Избегайте ветвлений внутри циклов
- Используйте inline функции
  - – O5 и –qipa делает это автоматически
- Избегайте aliasing
  - Один и тот же массив передается в функцию под разными именами
  - #pragma disjoint

```
#include <math.h>
#include <builtins.h>
__inline void aligned_ten_reciprocal_roots (double* x, double* f)
{
#pragma disjoint (*x, *f)
int i;
    __alignx (16, x);
    __alignx (16, f);
    for (i=0; i < 10; i++)
   f[i]= 1.0 / sqrt (x[i]);
}
```

```
extern
#ifdef __cplusplus
"builtin"
#endif
void __alignx (int n, const void *addr)
```

```
void reciprocal_roots (double *x, double *f, int n)
{
    /* are both x & f 16 byte aligned? */
    if ( ((((int) x) | ((int) f)) & 0xf) == 0) /* This could also be done as:
                                    if (((int) x % 16 == 0) && ((int) f % 16) == 0) */
   aligned_ten_reciprocal_roots (x, f, n);
   else
   ten_reciprocal_roots (x, f, n);
}
```

IBM pSeries 690 Regatta **

- Архитектура с общей памятью
  - 16 POWER4 64-bit процессоров @ 1.3GHz
  - 64 GB оперативной памяти
- Компиляция и управление задачами:
  - mpicc, mpiCC, mpif77, mpif90
  - mpisubmit -w 00:30:00 -n 16 myprog 0.01
  - llq, llcancel
- **http://www.regatta.cmc.msu.ru**