

High-Performance Computing Servers Seminar for Foreign Students

November 2, 2005
Hirofumi Amano
Computing and Communications Center
Kyushu University

1

About This Seminar

- This seminar is:
 - for those who are not good at Japanese;
 - designed to give a “*crash-course*” on how to use our High-Performance Computing Servers.
- This seminar is NOT:
 - a UNIX seminar;
 - a programming seminar;
 - a program tuning seminar.
- After this seminar:
 - you will be able to run your programs on our HPC Servers in interactive sessions or in batch jobs.

2

Other Useful Materials

- “*High-Performance Computing Servers (IBM p5) First-Step Guide*”
 - <http://isabelle.cc.kyushu-u.ac.jp/~amano/ccc/p5/>
- English documents provided by IBM and other software vendors
 - Chapter 2 of the above guide will be a good place to start your search.

3

How to Obtain Your Accounts

- For this seminar, you are given a temporary account.
- After this seminar:
 - Please consult your advisor (professor/associate professor) for your permanent account.
 - In some laboratories, your advisor may have already obtained accounts for the lab’s students.
- When you need a new account:
 - We recommend you to apply for your account with the help of your advisor.
 - This is because our HPC service is **NOT FREE** and the payment plan is not yours but your advisor’s.

4

Today's Schedule

- Basics
 - System Overview
 - Performance
 - Operation Details
 - Accessing the HPC Servers
- Compiling, Linking and Executing a Program
 - Compilation Commands and Their Options
 - Compilation Options for Parallel Programs
 - MPI parallel programs
- Batch Jobs for Large-Scale Computations
 - How the Batch System Works
 - Job Classes and Their Limits
 - Job Command Files
 - How to Monitor and Control Your Batch Jobs

Exercise

Exercise

5

Today's Schedule

- Basics
 - System Overview
 - Performance
 - Operation Details
 - Accessing the HPC Servers
- Compiling, Linking and Executing a Program
 - Compilation Commands and Their Options
 - Compilation Options for Parallel Programs
 - MPI parallel programs
- Batch Jobs for Large-Scale Computations
 - How the Batch System Works
 - Job Classes and Their Limits
 - Job Command Files
 - How to Monitor and Control Your Batch Jobs

6

Basics

7

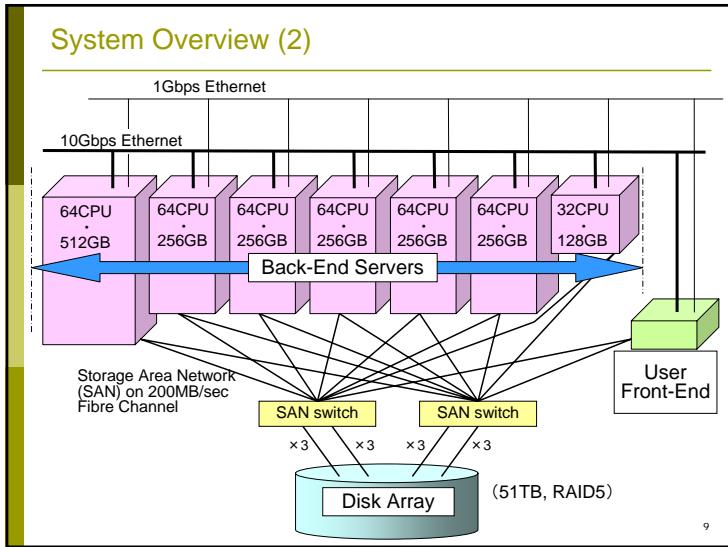
System Overview (1)

- High-Performance Computing Servers
IBM eServer p5 Model 595 (backend servers)
(AIX operating system)
 - POWER5 1.9GHz × 64CPU · 512GB memory ...1
 - POWER5 1.9GHz × 64CPU · 256GB memory ...5
 - POWER5 1.9GHz × 32CPU · 128GB memory ...1
- User Front-End
IBM eServer p5 Model 570
(AIX operating system)
 - POWER5 1.9GHz × 16CPU · 64GB memory ...1
- Disk Arrays
IBM TotalStorage FAST900
Storage Server
 - RAID-5 Effective Capacity: 51TB



IBM eServer p5 Model595

8



Performance (1)

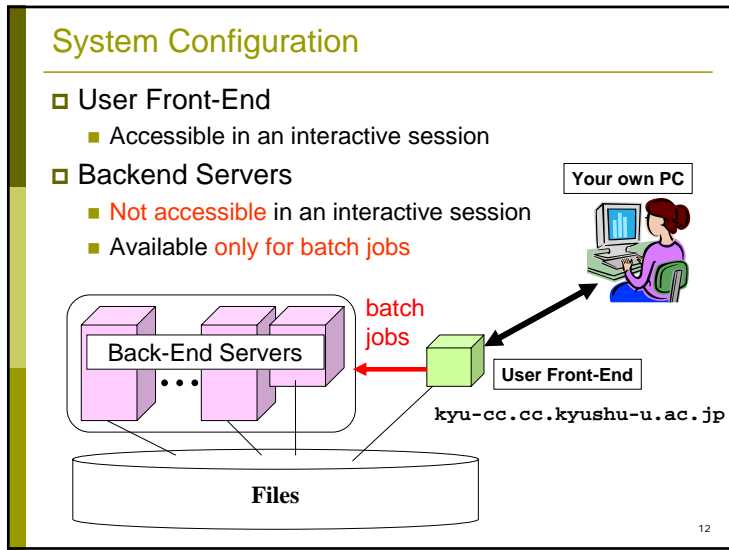
- Theoretical Peak Performance of a Single CPU: 7.6GFLOPS
 - 7,600,000,000 floating-point operations per second!
- Single-CPU Performance (SPEC Benchmark)

CPU	SPECint2000 base	SPECfp2000 base
IBM Power5 1.9GHz	1392	2585
Intel Itanium2 1.6GHz	1535	2675
Fujitsu SPARC64V 2.16GHz	1456	1808
Intel Pentium4 3.8GHz	1793	1976
- Parallel Performance (SPEC OMPL2001 base)

Machines	CPU	SPECompL2001 base
IBM p5 model 595	Power5 1.9GHz x 64	620741
SGI Altix 3700	Itanium2 1.6GHz x 64	507602
Fujitsu HPC2500	SPARC64V 1.3GHz x 128	262140

Performance (2)

- Single-CPU Performance Measured by More Realistic Programs
 - Sparse Matrix-Vector Product (100,000 dimensions, containing 1,000,000 non-zero elements)
 - IBM Power5 1.9GHz 0.009 sec
 - Intel Itanium2 1.5GHz 0.012 sec
 - Intel Xeon 3.2GHz 0.040 sec
 - Dense Matrix Product (2,000 x 2,000)
 - IBM Power5 1.9GHz 16.57 sec
 - Intel Itanium2 1.5GHz 11.23 sec
 - Intel Xeon 3.2GHz 34.61 sec



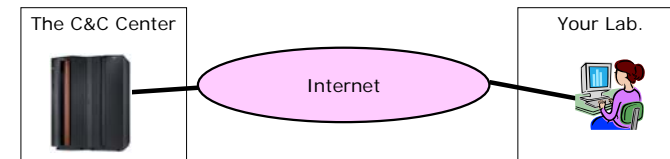
System Operation Details

- Operating Hours
 - In principle, 24 hours a day
 - Regular Maintenance: AM5:00~PM0:30 on Wednesdays
 - Service will be continued when no maintenance is required.
 - Examples of other service discontinuities (to be announced each time):
 - Winter holidays around the new year's day
 - A few days around the end of the fiscal year
- For more details on the system operations:
<http://www.cc.kyushu-u.ac.jp/scp/> (in Japanese)
- For questions or requests, contact:
request@cc.kyushu-u.ac.jp

13

Accessing the User Front-End (kyu-cc) (1)

- Logging in
 - **SSH** (Secure Shell) software must be installed.
 - Windows: Putty or TTSSH
 - MacOS: MacSSH
 - UNIX: OpenSSH
 - Conventional telnet access from outside **kyu-cc** is **disabled**.



14

Accessing the User Front-End (kyu-cc) (2)

- Uploading/downloading your files
 - **SFTP** (Secure FTP) capability is required.
 - Windows: WinSCP3
 - MacOS: MacSFTP
 - UNIX: `sftp` command (included in the OpenSSH package)
 - Conventional FTP access is **disabled**.
 - SCP access from outside **kyu-cc** is also **disabled**.

15

Download Sites for Client Software

- Windows
 - Putty
<http://www.chiark.greenend.org.uk/~sgtatham/putty/>
 - TTSSH
<http://sourceforge.jp/projects/ttssh2/>
 - WinSCP3
<http://winscp.net/>
- MacOS
 - MacSSH and MacSFTP
<http://pro.wanadoo.fr/chombier/>

16

In Your First Session...

- ❑ You must **change the initial password**.
 - In the first session, the system will ask you to change the password.
 - ❑ Enter the old (initial) password once, and enter your new password twice.
 - ❑ The session will be terminated after changing the password. Try the next and all the later sessions with the new password.
- ❑ File uploading/downloading is allowed **only after you change the initial password**.
- ❑ This restriction applies to both:
 - the temporary account for today;
 - the permanent account you will use later.

17

In Your Later Sessions...

- ❑ You must login to AIX again with your new password.
- ❑ Do not forget your password.

18

After You Logged in (TTSSH):

```
kyu-cc.cc.kyushu-u.ac.jp - i70036a@kyu-cc : /home/users02/i70036a VT
File Edit Setup Control Window Help
*****
*
* Welcome to AIX !
*
* Please refer to the following URL. (Sorry! Contents are only Japanese.)
* http://www.cc.kyushu-u.ac.jp/scp/system/hpc/p5.html
*
*
*****
[i70036a@kyu-cc ~ ]% :
```

Messages from the AIX.

prompt
(You can change this format for your taste.)

This is where you enter your command to the computer.

19

When You Finish Your Work of the Day...

- ❑ Do not forget to log out.
 - “**exit**” command will close the interactive session safely.

```
kyu-cc% exit ↵
```

This will make sure that no process remains after the session.
(Your batch jobs will stay intact even after logging off.)

20

Today's Schedule

- Basics
 - System Overview
 - Performance
 - Operation Details
 - Accessing the HPC Servers
- Compiling, Linking and Executing a Program
 - Compilation Commands and Their Options
 - Compilation Options for Parallel Programs
 - MPI parallel programs
- Batch Jobs for Large-Scale Computations
 - How the Batch System Works
 - Job Classes and Their Limits
 - Job Command Files
 - How to Monitor and Control Your Batch Jobs

21

Compiling, Linking and Executing a Program

First, we learn how to compile, link and execute a program in interactive sessions.

22

Sequential and Parallel Programs

- Sequential Programs:
 - Designed and coded for a single CPU
 - Executable on a single CPU
- Parallel Programs:
 - Executable codes on multiple CPUs
 - Methods to obtain executable parallel code:
 - Automatic Parallelization (no source modification required)
 - OpenMP (slight modification required)
 - MPI (extensive modification required)
 - Hybrid (auto-parallel + MPI, OpenMP + MPI)

See Appendix for more details.

To exploit the potential (capacity and performance) of our HPC Servers, *parallel programs* are important.

23

Basic Compilation Commands

Language	Sequential (single CPU)	Auto-Parallel/ OpenMP	MPI	Hybrid
C	cc	cc_r	mpcc	mpcc_r
Fortran77	f77	f77_r	mpf77	mpf77_r
Fortran90	f90	f90_r	mpf90	mpf90_r

- Other Commands
 - x1c ANSI C89-Compliant C Programs
 - x1f90 Fortran90 programs having an extension ".f"
 - x1c C++ Programs
- A dedicated command for each parallelization method

24

Compiling a Sequential Program

Compilation Command Format

```
compile_command options source_program ↵
```

Samples

- The compilation command will depend on your choice of the programming language (see Slide #24).

```
kyu-cc% f90 example.f90 ↵
```

This will create a file "a.out" for the executable code.

```
kyu-cc% f90 -o example example.f90 ↵
```

This option will change the name of the executable code file into "example".

25

Executing a Sequential Program

Execution Command Format

```
location_of_executable_file ↵
```

Samples

```
kyu-cc% ./a.out ↵
```

This will execute the file "a.out" in the current directory.

```
kyu-cc% ./example ↵
```

This will execute the file "example" in the current directory.

"." indicates the current directory.

"/example" means "the file example in the current directory".

26

Compilation Options

Directions given to the compiler for controlling the compilation specifics

- Some options are common to the programming languages.
- Other options are specific to individual languages.

Each option starts with "-".

- Some options may have subsequent parameters.
- Examples:

-c

-o filename

27

Basic Compilation Options for Fortran

-c	Create an object file instead of an executable file.
-o filename	Store the output (executable or object) into the file specified by <i>filename</i> , instead of the default (*.o or a.out).
-qfree	Compile the file as a free-format Fortran source program.
-qfixed	Compile the file as a fixed-format Fortran source program.
-O	Apply basic optimizations only.
-O3	Apply deeper optimizations such as changing the execution order of operations. This may cause some side effects.
-O4	Apply further optimizations in addition to those caused by -O3.
-O5	Try the deepest optimizations.
-gstrict	(Together with optimization option -O3, -O4, or -O5) Create an executable/object code which preserves the original execution order of operations specified in the source.

28

Basic Compilation Options for C/C++

<code>-c</code>	Create an object file instead of an executable file.
<code>-o filename</code>	Store the output (executable or object) into the file specified by <i>filename</i> , instead of the default (*.o or a.out).
<code>-lm</code>	Link mathematical functions in math library. This option must be specified at the end of the command line.
<code>-O</code>	Apply basic optimizations only.
<code>-O3</code>	Apply deeper optimizations such as changing the execution order of operations. This may cause some side effects.
<code>-O4</code>	Apply further optimizations in addition to those caused by <code>-O3</code> .
<code>-O5</code>	Try the deepest optimizations.
<code>-qstrict</code>	(Together with optimization option <code>-O3</code> , <code>-O4</code> , or <code>-O5</code>) Create an executable/object code which preserves the original execution order of operations specified in the source.

29

Most Recommended Optimization Options

- In most cases, the following compiler options are expected to give you a sufficient performance improvement.

```
-O3 -qarch=pwr5 -qtune=pwr5
```

- General Notes on Optimization

- A higher (deeper) optimization requires a longer compilation time.
`-O3 < -O4 < -O5`
- A higher optimization may not always give you a faster executable code.
- Optimization may cause side effects in the computation results.
 - To avoid such side effects, use “`-qstrict`” option.

30

Automatic Parallelization Option

- To make the compiler create a parallel executable code from a sequential source program:

```
-qsmp=auto
```

- The compilation command must be one with “_r” at the end of the command name.

```
kyu-cc% f90_r -qsmp=auto -o example example.f90
```

```
kyu-cc% f77_r -qsmp=auto -o example example.f
```

```
kyu-cc% cc_r -qsmp=auto -o example example.c
```

31

OpenMP Compilation Option

- To make the compiler create a parallel executable code from a OpenMP source program:

```
-qsmp=omp
```

- The compilation command must be one with “_r” at the end of the command name.

```
kyu-cc% f90_r -qsmp=omp -o example example.f90
```

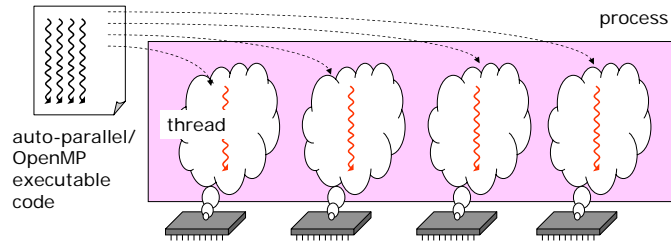
```
kyu-cc% f77_r -qsmp=omp -o example example.f
```

```
kyu-cc% cc_r -qsmp=omp -o example example.c
```

32

Executing a Thread-Parallel Program (1)

- A parallel program made by auto-parallel or OpenMP runs on multiple *threads* within a single process.



If there are enough number of CPUs available, each thread runs on a CPU. Otherwise, some of them may run on the same CPU.

33

Executing a Thread-Parallel Program (2)

- The number of threads:

- To be specified by an environment variable "OMP_NUM_THREADS"

- In csh or tcsh:

```
kyu-cc% setenv OMP_NUM_THREADS 4
```

- In sh, bash, ksh:

```
kyu-cc% export OMP_NUM_THREADS=4
```

- The default value is the number of CPUs installed in the computer.

- To display the current value:

```
kyu-cc% echo $OMP_NUM_THREADS
```

The syntax depends on the command language interpreter (shell).

34

Executing a Thread-Parallel Program (3)

- The value of "OMP_NUM_THREADS" will be preserved until you set it again or you log out.
- To execute a parallel executable code by auto-parallel or OpenMP:
 - Just type the file name.

```
kyu-cc% ./example
```

35

MPI Compilation

- The compilation command must be one starting with "mp" at the head of the command name.

```
kyu-cc% mpf90 -o example example.f90
```

```
kyu-cc% mpf77 -o example example.f
```

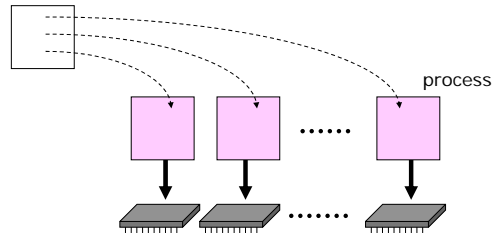
```
kyu-cc% mpcc -o example example.c
```

36

Executing MPI Programs (1)

- A parallel program made with MPI runs on multiple *processes*.

MPI executable program



37

Executing MPI Programs (2)

- The number of processes:
 - In interactive sessions, it must be specified by “-procs” option at each execution.

```
kyu-cc% ./example -procs 4
```

Important Notes:

- (1) This approach is quite different from thread-parallel cases.
- (2) This syntax **DOES NOT APPLY** to batch jobs. (See “JCF Sample (3): an MPI Program” in Slide #55 for more details.)

38

Charges

- Our HPC service is **NOT FREE** of charge.
 - The choice of the payment plan is not yours but your advisor's.
 - Please refer to Chapter 3 of: <http://isabelle.cc.kyushu-u.ac.jp/~amano/ccc/p5/> for more details of our charging system.
 - However, **today's exercise is free**.
- **Warning:**
 - The current CPU time charge for shared resource plans is based on the **total CPU time of the program**.
 - This means that **most parallel programs costs more money than a sequential version**.
 - You must carefully consider the tradeoff between the increased cost and the improved response.

39

Exercise (1): Interactive Sessions

- Accessing **kyu-cc**
 - Logging in Windows
 - Logging in AIX
 - Changing your initial password
- Preparing sample files
 - Copying the sample file package and unpacking it
- Compiling a sequential program
- Automatic parallelization
- OpenMP: compilation and execution
- MPI: compilation and execution

See the separate instructions for more details.

40

Today's Schedule

- Basics
 - System Overview
 - Performance
 - Operation Details
 - Accessing the HPC Servers
- Compiling, Linking and Executing a Program
 - Compilation Commands and Their Options
 - Compilation Options for Parallel Programs
 - MPI parallel programs
- Batch Jobs for Large-Scale Computations
 - How the Batch System Works
 - Job Classes and Their Limits
 - Job Command Files
 - How to Monitor and Control Your Batch Jobs

41

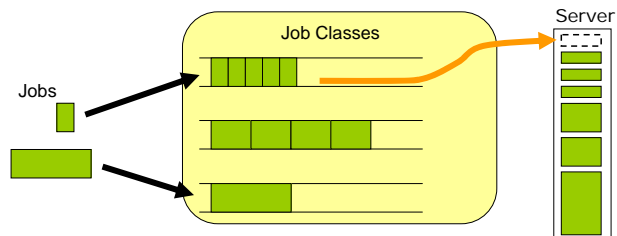
Batch Jobs for Large-Scale Computations

Now, we are ready to carry out a large computation which cannot be processed in interactive sessions.

42

How the Batch System Works

- Users submit their request (**job**).
 - The batch job scheduler accepts the submitted job, but may not start it immediately.
- The scheduler starts a job when the required resource becomes available.
 - Jobs are stored in **job classes** which characterize the size of the required resource.



43

Why Batch, Not Interactive?

- In interactive sessions:
 - Some commands cannot be executed when the remaining computer resource is not enough.
 - Such a command will be rejected.
 - You cannot tell when the resource will become available.
 - Perhaps you do not wish to keep typing your command.
- “Reservation” mechanism is a good solution for this problem.
 - The system will automatically start the execution of your command when the required resource becomes available.

44

When to Use the Batch System

- When you need a large computer resource:
 - many CPUs
 - a large memory
 - a long execution time
- In an interactive session (on `kyu-cc`):
 - The maximum memory is limited to 1GB.
 - The maximum CPU time is limited to 1 hour.
 - (Every user share the 16 CPUs installed in the user front-end with the other users.)
- All requests exceeding this limit will be terminated automatically.
 - you must execute them on the **back-end servers** with the help of the batch system.

45

Job Classes

- Classify the user requests by the resource limits
 - the number of CPUs (#CPUs)
 - memory size
 - the number of processes (#processes)
 - the execution time
- Scheduling Policy
 - In the same job class, each job will be executed in the first-in, first-out order.
 - When we look at multiple job classes, a job in one class may pass some waiting jobs in other classes.
 - Smaller jobs are likely to run before larger ones.

46

Resource Limits

The total CPU time for a single process

class	memory	time	#processes	#CPUs	description
interactive	1GB	1 hour	---	---	All users share the 16 CPUs.
e1	3GB	1 week	1	1	For OpenMP or auto-parallel jobs
e2	6GB	1 week	1	2	
e4	12GB	1 week	1	4	
e8	24GB	1 week	1	8	
e16	48GB	1 week	1	16	
ms	3GB/process	1 week	16	16	for MPI jobs
mm	6GB/process	1 week	8	16	
ml	12GB/process	1 week	4	16	

The elapsed time

47

Cautions

- The batch job classes listed in the table are those for shared use.
 - Your laboratory may have made a contract on the **exclusive resource plan**.
 - In that case, your lab has the dedicated job classes assigned **only for that lab**.
 - Please consult your advisor for more details.
- The numbers of CPUs listed in the table:
 - Guaranteed for a single batch job once it starts running.
 - Will stay granted to the running job even when the system becomes heavily loaded.
- A job can declare a greater number of threads.
 - It may be granted more CPUs when available.
 - However, it may share extra CPUs with other jobs when the system is busy.

48

Other Limits

- ❑ The maximum number of jobs which can be executed for a single user:
8
- ❑ You can submit as many jobs as you like, but only 8 of them can run at one time.

49

Job Command File (JCF)

- ❑ A text file which describes the sequence of commands to execute in the job
 - You can create your JCF with a text editor on the front-end (kyu-cc).
 - You can also edit one with any other text editor installed on your PC and transfer it to kyu-cc.
- ❑ A JCF looks similar to a shell script file:
 - Containing some control statements at the top of the file
 - Containing a sequence of commands to be executed.

50

JCF (Differences from FUJITSU VPP5000)

- ❑ A JCF looks just like an ordinary shell script, but it is NOT.
 - ❑ On VPP, a job is described in a plain shell script file.
- ❑ The target job class must be explicitly specified in the JCF.
 - ❑ On VPP, it can be specified in a command line option at job submission time.
- ❑ The input and the output files must be specified explicitly and properly in the JCF.
 - ❑ On VPP, those files are automatically generated.
 - ❑ Improper setting will cause the complete loss of computing results or the accidental destruction of the previous results of the same JCF.
- ❑ The initial working directory is the directory when the JCF is submitted.
 - ❑ On VPP, it is the user's home directory.

51

JCF Syntax

- ❑ Control lines starting with “# @” at the top of a JCF
 - Specifying the configuration of the job

```
#!/usr/bin/csh
# @ class = job_class_name
# @ output = file_to_store_standard_output
# @ error = file_to_store_standard_error
# @ queue
command_to_execute_1
command_to_execute_2
:
```

52

JCF Sample (1): a Sequential Program

- Choose the job class according to the memory requirement.
- “\$(jobid)” in a JCF will be automatically substituted by the job ID.
 - Standard output and standard error can be a single (same) file.

```
#!/usr/bin/csh

# @ class = e1
# @ output = test1.o$(jobid)
# @ error = test1.e$(jobid)
# @ queue
./a.out
```

This is a good practice to avoid accidentally destroying the result of the previous execution of the same JCF.

53

JCF Sample (2): Auto-Parallel, OpenMP


- The number of threads must be specified **before** the execution of parallel programs.

```
#!/usr/bin/csh

# @ class = e8
# @ output = test2.o$(jobid)
# @ error = test2.e$(jobid)
# @ queue
setenv OMP_NUM_THREADS 4
./a.out
```

54

JCF Sample (3): an MPI Program

- An MPI job involves multiple processes.
 - To execute this kind of parallel jobs, “job_type = parallel” must be declared in the JCF.
 - “total_tasks” value must be fixed to determine the number of processes.
 - Execution option “-procs” will be ignored.  cf. Slide #38

```
#!/usr/bin/csh

# @ class = mm
# @ job_type = parallel
# @ total_tasks = 4
# @ output = mpi.jcf.o$(jobid)
# @ error = mpi.jcf.e$(jobid)
# @ queue
./a.out
```

55

JCF Sample (4): Gaussian03

- Allocate the scratch files to be used by Gaussian in the local file system of the back-end server instead of the front-end.
 - To achieve better I/O performance
 - Must be deleted after the execution

```
#!/usr/bin/csh

# @ class = e1
# @ output = test4.o$(jobid)
# @ error = test4.e$(jobid)
# @ queue
setenv GAUSS_SCRDIR /work/users/${LOADL_STEP_OWNER}/${LOADL_JOB_NAME}
g03 test
/usr/bin/rm -rf /work/users/${LOADL_STEP_OWNER}/${LOADL_JOB_NAME}/*
```

56

Other Useful Keywords in JCF

keywords	description
initialdir	the initial directory (default: where the directory the JCF is submitted)
notify_user	destination of the notification e-mail (default: <i>your_ID</i> @kyu-cc.cc.kyushu-u.ac.jp)
notification	timing of e-mail notification (default: complete) <ul style="list-style-type: none"> complete Notify when the job is finished. start Notify when the job is started. error Notify when the job is terminated abnormally. always Notify for all the above occasions. never Never notify.

57

Notification by e-Mails

- We recommend you to enable the mail forwarding.
 - The default destination of notification mails is:
your_ID@kyu-cc.cc.kyushu-u.ac.jp
 - You cannot read the mails unless you log in *kyu-cc*.
 - Instead, the mails can be redirected to your office mail account.
 - Just write your address in “.forward” file in the home directly.

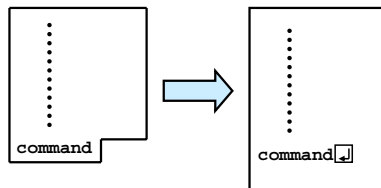
```
kyu-cc% echo your_address > .forward
```

- This will save you specifying “**notify**” option in every JCF.

58

Other Tips for JCF

- The output and error files will be overwritten each time, if the file names are fixed.
 - If the file names have “\$(jobid)” as suffix, the accidental loss of the previous execution will be avoided.
- The end line of a JCF must be a blank line.



59

How to Monitor and Control Your Jobs

- Job Submission:
llsubmit
- Job Status Monitoring
qps
- Job Canceling
llcancel

60

Job Submission: `llsubmit`

Basic Command Usage:

```
llsubmit JCF_file_name
```

```
kyu-cc% llsubmit test.jcf
```

61

Job Status Monitoring: `qps`

Basic Command Usage:

```
qps option
```

This will display the list of:

- all the jobs submitted by the user (without option)
- all the running jobs (for “-a” option)
- all the waiting jobs (for “-q” option)

R: Running
I: Idle
NQ: Not Queued

```
kyu-cc.14524.0 k70043a 3/8 17:57 R 50 e4 kyu-cc-g 00:10:43
```

job ID	user ID	submission date/time	status	priority	job class	consumed CPU time
kyu-cc.14524.0	k70043a	3/8 17:57	R	50	e4 kyu-cc-g	00:10:43

- When the list contain too many lines:

```
kyu-cc% qps -a | less
```

62

Job Canceling: `llcancel`

Basic Command Usage:

```
llcancel job_ID
```

```
llcancel job_number
```

- job ID format: `kyu-cc.12345.0`
- job number format: `12345`
- job ID (job number) can be obtained by “`qps`” command.

```
kyu-cc% lllcancel 12345
```

Both usages give you the same effect.

```
kyu-cc% lllcancel kyu-cc.12345.0
```

63

Exercise (2): Batch Jobs

- Job Command Files (JCFs)
- Job Submission
- Job Status Monitoring
- Job Canceling

See the separate instructions for more details.

64

Appendix

- Categories of Basic Parallelization Approaches (Slide #66)
- Using Numerical Libraries (Slide #76)

65

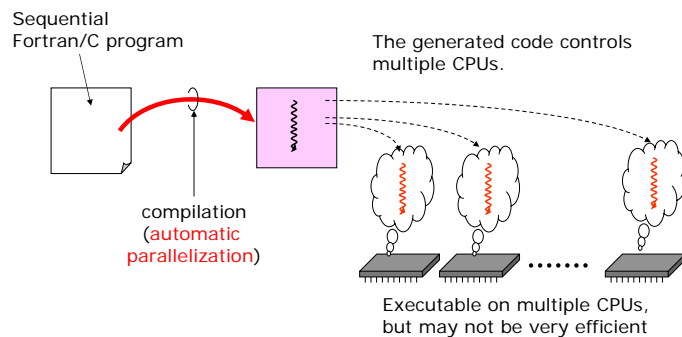
Categories of Basic Parallelization Approaches

- Automatic Parallelization
 - no source program modification required
 - thread-parallel execution
- OpenMP
 - slight source program modification required
 - thread-parallel execution
- MPI
 - extensive source program modification required
 - process-parallel execution
- Hybrid
 - auto-parallel + MPI, or, OpenMP + MPI

66

Automatic Parallelization

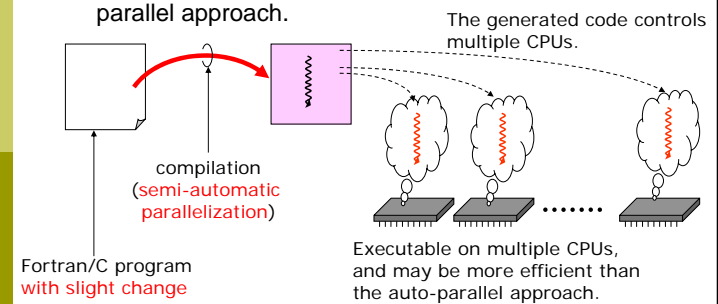
- No source program modification required
 - A sequential program is automatically parallelized by the compiler, but *with a limited capability*.



67

OpenMP

- Slight source program modification required
 - The executable code is directly generated by an OpenMP-capable compiler.
 - The executable code maybe more efficient than auto-parallel approach.



68

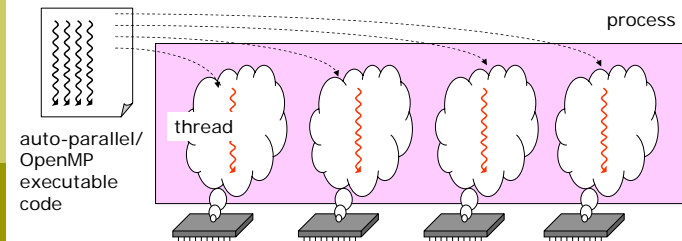
OpenMP Resource

- For more details, please refer to other resources such as:
 - The OpenMP Architecture Review Board Web Site: <http://www.openmp.org/>
 - Rohit Chandra, Ramesh Menon, Leo Dagum, David Kohr, Dror Maydan, Jeff McDonald: *"Parallel Programming in OpenMP"*, Morgan-Kaufman, 2000. (ISBN: 1-55860-671-8)
 - Michael J. Quinn: *"Parallel Programming in C With MPI and OpenMP"*, Mcgraw-Hill College, 2003. (ISBN: 0072822562)

69

Execution of a Thread-Parallel Program

- A parallel program made by auto-parallel or OpenMP runs on multiple *threads* within a single process.

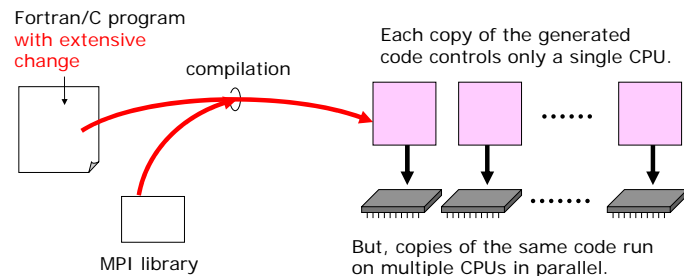


If there are enough number of CPUs available, each thread runs on a CPU. Otherwise, some of them may run on the same CPU.

70

MPI

- Extensive source modification required
 - The modified version is no longer a sequential program.
 - Programmers must control the communication and synchronization between processes.



71

MPI Resource

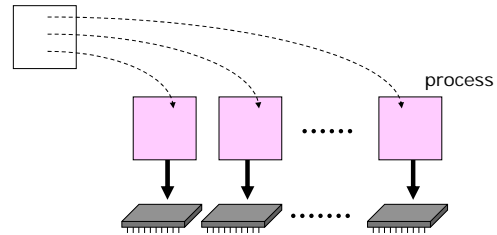
- For more details, please refer to other resources such as:
 - Peter Pacheco: *"Parallel Programming With MPI"*, Morgan Kaufmann Pub., 1996. (ISBN: 1558603395)
 - William Gropp, Ewing Lusk, Anthony Skjellum: *"Using MPI: Portable Parallel Programming With the Message-Passing Interface (2nd Edition, Scientific and Engineering Computation Series)"*, MIT Press, 1999. (ISBN: 0262571323)
 - Michael J. Quinn: *"Parallel Programming in C With MPI and OpenMP"*, Mcgraw-Hill College, 2003. (ISBN: 0072822562)
 - <http://www.mpi-forum.org/>

72

Execution of a Process-Parallel Program

- A parallel program made with MPI runs on multiple *processes*.

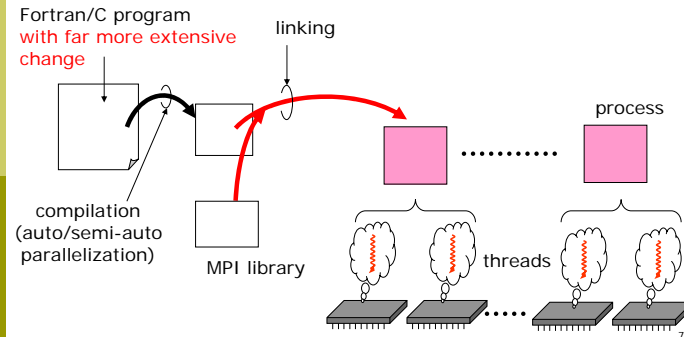
MPI executable program



73

Hybrid Approach

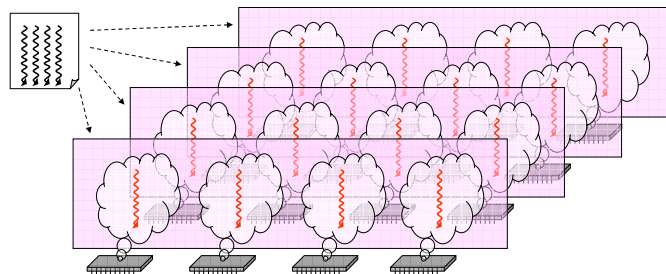
- The most complex approach
 - Automatic Parallelization + MPI
 - OpenMP + MPI



74

Execution of a Hybrid-Parallel Program

- A parallel program made by the hybrid approach runs on *multiple threads on multiple processes*.



75

Using Numerical Libraries (1)

- ESSL
 - Compile the source program with the following option:
 - lessl ... for the sequential version of ESSL
 - lesslsmpl ... for the parallel version of ESSL (must be compiled with “_r” commands)
 - IMSL/Fortran Library
 - \$F90FLAGS (for compilation) and \$LINK_F90 (for link-editing) must be set properly.
- ```
kyu-cc% source /usr/appl/CTT6.0/ctt/bin/cttsetup.csh
```
- IMSL/C Library
    - \$CFLAGS (for compilation) and \$LINK\_CNL (for link-editing) must be set properly.

76

## Using Numerical Libraries (2)

---

### □ NAG

- Compile the source program with the following option:
  - lnag ... for static linking
  - lnag\_sh ... for dynamic linking
  - lnag\_use\_essl -lessl ...use it it with BLAS in ESSL

### □ LAPACK

- Compile the source program with the following option:
  - llapack -lessl ... to be used with the sequential version of ESSL
  - llapack\_smp -lesslsmp ... to be used with the parallel version of ESSL (must be compiled with “\_r” commands)

77

## For More Information...

---

- Our English materials are not provided well. We apologize to you for inconvenience.
- If you have any question, however, please do not hesitate to contact:  
[amano@cc.kyushu-u.ac.jp](mailto:amano@cc.kyushu-u.ac.jp)

78

---

***Thank you!***

79