A Brief Introduction to Parallel Computing

Zhang, Honglin

Department of Physics, Tsinghua University

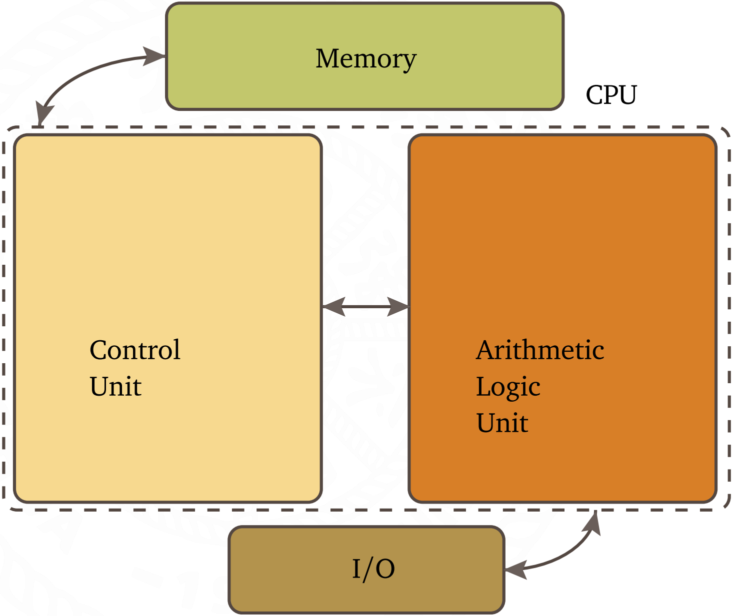
June 25, 2012

Prologue

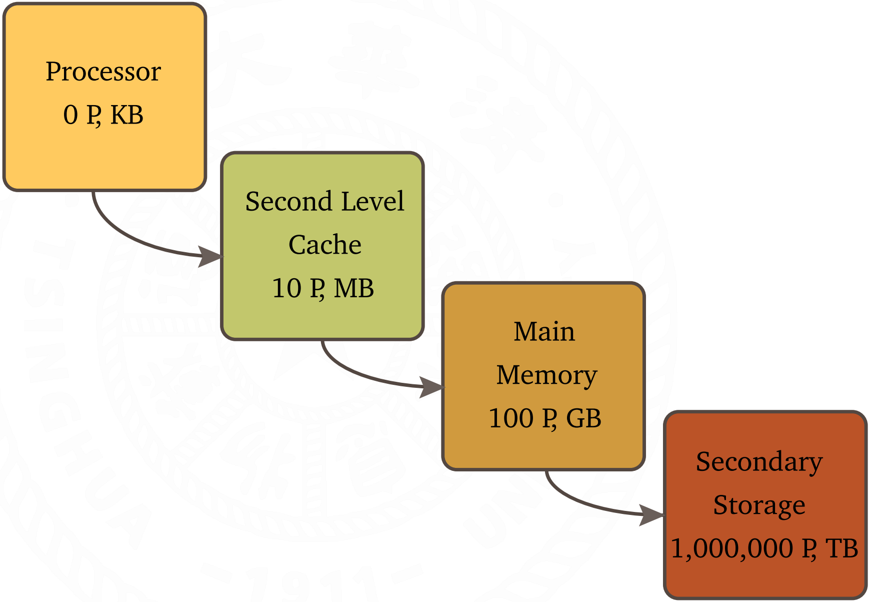
Why Parallel Computing?

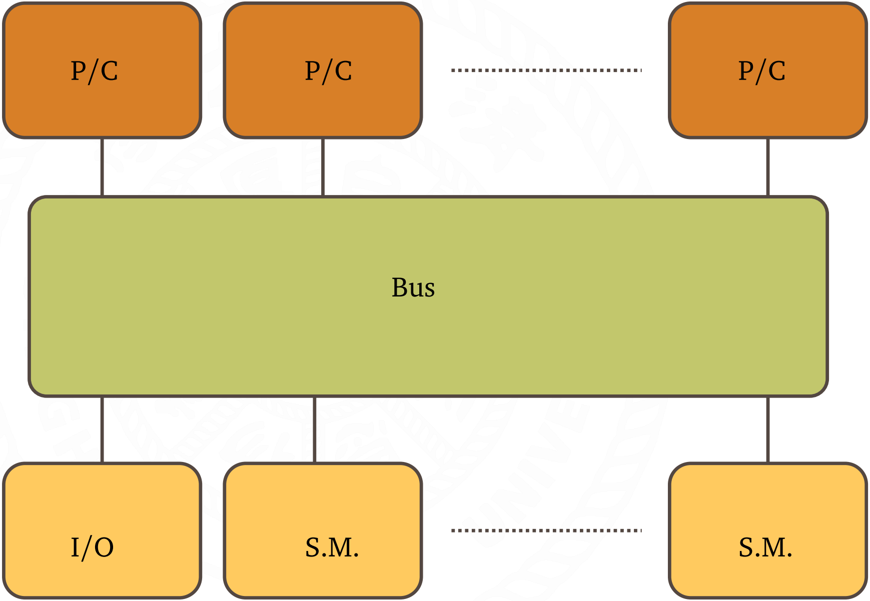
Quick Review

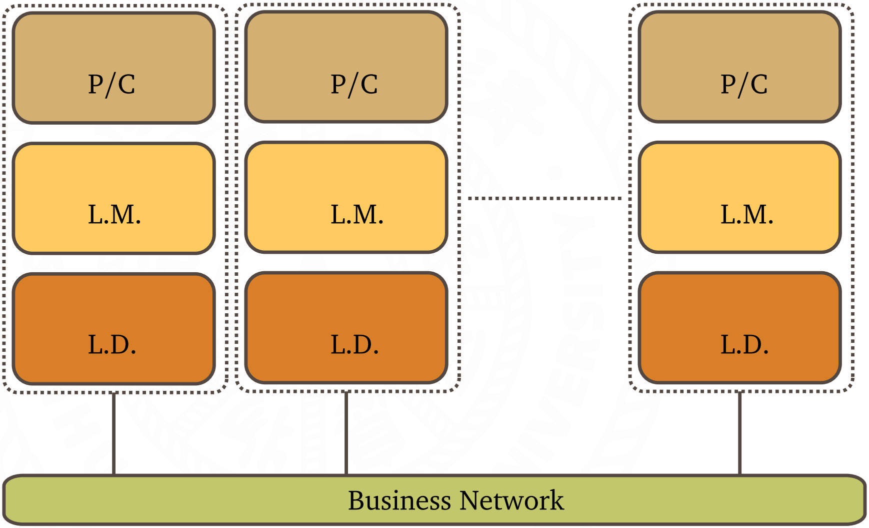
Introduction to Hardware


Memory
CPU
Control Unit
Arithmetic Logic Unit
I/O

Device	Pro.	Sec. Lev. Cac.	Main Mem.	Sec. Sto.
CPU Period	0	10	100	1,000,000
Speed	10GB/s	2GB/s	1GB/s	10MB/s
Size	KB	MB	GB	TB


Processor
0 P, KB
Second Level
Cache
10 P, MB
Main
Memory
100 P, GB
Secondary
Storage
1,000,000 P, TB


P/C
P/C
P/C
Bus
I/O
S.M.
S.M.


P/C
P/C
P/C
L.M.
L.M.
L.M.
L.D.
L.D.
L.D.
Business Network

Before we hit the road,

Processes vs. Threads

Processes

Resource Holder
Independent Execution
Programs Loaded as Processes by OS

Processes

Resource Holder
Independent Execution
Programs Loaded as Processes by OS

Threads

(Master) Thread at Program Entry Point
More Threads Created by Existing Ones Possible
Separate Stack for each thread
Shared Code and Data Segments

Why Threads?

- Efficient Data Sharing

Why Threads?

- Efficient Data Sharing
- High Multi-core Performance

Typical Parallel Models I.

Shared Memory Model

Shared Memory Model

- Unified Address Space

Shared Memory Model

- Unified Address Space
- Asynchronous Execution

Shared Memory Model

- Unified Address Space
- Asynchronous Execution
- Able to Explicitly Synchronize

Shared Memory Model

- Unified Address Space
- Asynchronous Execution
- Able to Explicitly Synchronize
- Local Variables along with Shared Variables

Shared Memory Model

- Unified Address Space
- Asynchronous Execution
- Able to Explicitly Synchronize
- Local Variables along with Shared Variables
- Threads Created and Destroyed Dynamically

A Typical Terrible Example

```
static int s=0;
```

Thread 1

```
reg0=f(A[i]);
reg1=s;
reg1=reg1+reg0;
s=reg1;
```

Thread 2

```
reg0=f(A[i]);
reg1=s;
reg1=reg1+reg0;
s=reg1;
```

A Typical Terrible Example

static int s=0;

Thread 1

```
reg0=f(A[i]);
reg1=s;
reg1=reg1+reg0;
s=reg1;
```

Thread 2

```
reg0=f(A[i]);
reg1=s;
reg1=reg1+reg0;
s=reg1;
```

Wait a moment

What value will “s” take?

Eureka!

Here is the problem! There are more than one threads visiting the variable "s" at the same moment.

Eureka!

Here is the problem! There are more than one threads visiting the variable "s" at the same moment.

Remember

This is very important!

A Typical Good Example

static int s=0;
static lock lk;

Thread 1

```
local_s1=0;
for i=0, n/2-1
    local_s1=local_s1+f(A[i]);
lock(lk);
s=s+local_s1;
unlock(lk);
```

Thread 2

```
local_s2=0;
for i=0, n/2-1
    local_s2=local_s2+f(A[i]);
lock(lk);
s=s+local_s2;
unlock(lk);
```

OpenMP

To implement shared memory model, we usually adopt OpenMP.

To implement shared memory model, we usually adopt OpenMP.

Tips

OpenMP is not a language, but a compiler directive!

To implement shared memory model, we usually adopt OpenMP.

Tips

OpenMP is not a language, but a compiler directive!

Supported by:

- C/C++/Fortran
- Sun Compiler/GNU Compiler/Intel Compiler

Typical Parallel Models II.

Message Passing Model

Argonne
NATIONAL LABORATORY

Message Passing Model

- Separate Address Space

Message Passing Model

- Separate Address Space
- Explicit Send/Receive Pair

Message Passing Model

- Separate Address Space
- Explicit Send/Receive Pair
- Explicitly Synchronize as Default

Message Passing Model

- Separate Address Space
- Explicit Send/Receive Pair
- Explicitly Synchronize as Default
- Process Number Decided Before Launch

Message passing involves message sending and receiving.

Message passing involves message sending and receiving.

Two typical forms of communication:

- Blocking, e.g. **Telephone**

Message passing involves message sending and receiving.

Two typical forms of communication:

- Blocking, e.g. **Telephone**
- Non-blocking, e.g. **Post Office**

Emm,

Feel the difference?

A Typical Terrible Example

Process 1

```
xlocal=A[1];
send xlocal, proc2;
receive xremote, proc2;
s=xlocal+xremote
```

Process 2

```
xlocal=A[2];
send xlocal, proc1;
receive xremote, proc1;
s=xlocal+xremote
```

A Typical Terrible Example

Process 1

xlocal=A[1];
send xlocal, proc2;
receive xremote, proc2;
s=xlocal+xremote

Process 2

xlocal=A[2];
send xlocal, proc1;
receive xremote, proc1;
s=xlocal+xremote

Wait a moment

Will there be an interlocking?

Eureka!

Send & Receive are blocking operation. The sender will wait silently until the message is received by a receiver.

Eureka!

Send & Receive are blocking operation. The sender will wait silently until the message is received by a receiver.

Remember

This is very important!

A Typical Good Example

Process 1

```
xlocal=A[1];
send xlocal, proc2;
receive xremote, proc2;
s=xlocal+xremote;
```

Process 2

```
xlocal=A[2];
receive xremote, proc1;
send xlocal, proc1;
s=xlocal+xremote;
```

A Typical Good Example

Process 1

```
xlocal=A[1];
send xlocal, proc2;
receive xremote, proc2;
s=xlocal+xremote;
```

Process 2

```
xlocal=A[2];
receive xremote, proc1;
send xlocal, proc1;
s=xlocal+xremote;
```

Of course, we may also implement this in a non-blocking way...

A Typical Good Example

Process 1

xlocal=A[1];
send xlocal, proc2;
receive xremote, proc2;
s=xlocal+xremote;

Process 2

xlocal=A[2];
receive xremote, proc1;
send xlocal, proc1;
s=xlocal+xremote;

Of course, we may also implement this in a non-blocking way...

However, non-blocking is not always reliable.

MPI

To implement message passing model, we usually adopt MPI.

MPI

To implement message passing model, we usually adopt MPI.

Tips

MPI is a library, not a language!

MPI

To implement message passing model, we usually adopt MPI.

Tips

MPI is a library, not a language!

Supporting:

- C/C++/Fortran

Want to know more?

More About OpenMP

Obtain the number of threads:

C/C++: int omp_get_num_threads(void)

Fortran: integer function omp_get_num_threads()

Obtain the number of threads:

C/C++: int omp_get_num_threads(void)
Fortran: integer function omp_get_num_threads()

Obtain the thread number

C/C++: int omp_get_thread_num(void)
Fortran: integer function omp_get_thread_num()

Notice thread 0 is the master!

Obtain the number of processors

C/C++: int omp_get_num_procs(void)

Fortran: integer function omp_get_num_procs()

Obtain the number of processors

C/C++: int omp_get_num_procs(void)
Fortran: integer function omp_get_num_procs()

Set the number of threads

C/C++: void omp_set_num_threads(int number_threads)
Fortran: subroutine omp_set_num_threads(number_threads)

More for you to find

https://computing.llnl.gov/tutorials/openMP/

http://openmp.org/wp/openmp-specifications/

Still More?

More About MPI

MPI defines a library for C, Fortran and Java. MPI's definition includes no specific features of any manufacturer, OS or device.

MPI defines a library for C, Fortran and Java. MPI's definition includes no specific features of any manufacturer, OS or device.

Especially, MPICH environment is broadly accepted.

MPICH Download

http://www.mcs.anl.gov/research/projects/mpich2

Compile in MPICH

- mpicxx/mpicc/mpif77/mpif90
- -cc=icc/gcc designates a specific compiler

Launch a MPI program

mpiexec -n N routine

mpiexec -host HOST_1 -n N routine_1 : -host HOST_2 -n M routine_2

Message Envelope for Sender

MPI_Send(address,count,datatype,destination,tag,communicator)

Description

address: data pointer
count: length of datatype
datatype: unified data type in MPI
destination: ID of process receiver
tag: for programmer use
communicator: ID of communication group

Message Envelope for Receiver

MPI_Recv(address,count,datatype,source,tag,communicator,status)

Description

status: a pointer of datatype MPI_Status, which record the information of sending and receiving

Broadcast

MPI_Bcast(address,count,datatype,root,comm)

Scatter

For Root:

MPI_Scatter(send_address,send_count,send_datatype,root,comm)

For others:

MPI_Scatter(recv_address,recv_count,recv_datatype)

Gather

For Root:

MPI_Gather(recv_address,recv_count,recv_datatype,root,comm)

For others:

MPI_Gather(send_address,send_count,send_datatype)

Barrier

MPI_Barrier(comm)

Description

Let processes in the same communicator wait for each other.

More for you to find

http://www.mcs.anl.gov/research/projects/mpi/

Not Finished Yet?

About GPU Programming

So far, we have not mentioned CUDA programming yet.

CUDA is so *unfriendly* that I would not address here.

More for you to find

http://www.nvidia.com/object/cuda_home_new.html

Advanced Tips for HPC?

- Profiling tools, e.g. VTune.

Advanced Tips for HPC?

- Profiling tools, e.g. VTune.
- Find senior programmers for help.

Advanced Tips for HPC?

- Profiling tools, e.g. VTune.
- Find senior programmers for help.
- Parallelize your program only when you have to.

Advanced Tips for HPC?

- Profiling tools, e.g. VTune.
- Find senior programmers for help.
- Parallelize your program only when you have to.

Q&A

Any Question?

Epilogue

Thanks for your Attention!