The Performance Analysis of Portable Parallel Programming Interface MpC for SDSM and pthread

Workshop DSM2005

CCGrig2005

Seikei University

Tokyo, Japan

midori@st.seikei.ac.jp

Hiroko Midorikawa

Outline

1. Meta Process Model
2. MpC Language
3. MpC program Execution Performance

- on clusters
  - using SDSMs
  - MpC vs. OpenMP
- on shared memory parallel machines
  - using pthread
  - MpC vs. UPC, OpenMP

4. Programming Productivity - Program line counts

Background

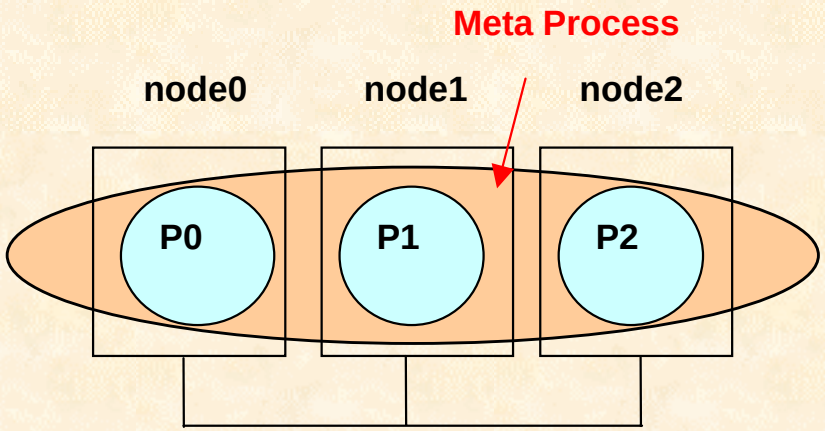
- Parallel Programming Model on clusters
  - MPI
    - Explicit description by programmers, tunable
    - Bothersome/Tedious message passing statements
    - Poor Readability
  - OpenMP
    - Insufficient API for distributed data mapping on nodes
    - Automatic insertion of redundant memory consistency synchronization

Meta Process Model

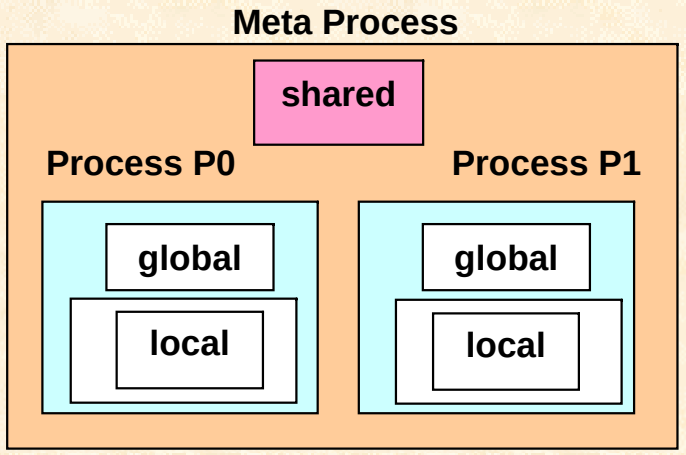
Hierarchical Shared Memory Model

- Explicit parallelism description paradigm
  - Meta Process :A group of processes to cooperate to achieve a single application, One execution entity for a user

- Explicit local/shared data distinction Process (not thread) Model
  - *Shared* data accessed by all processes in a Meta process



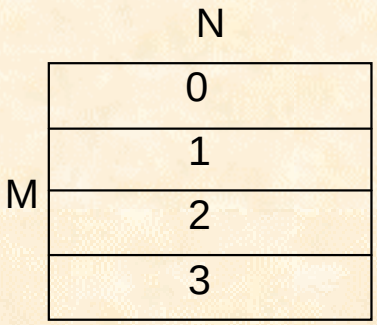
Meta Process and Processes for clusters



Hierarchical data scope

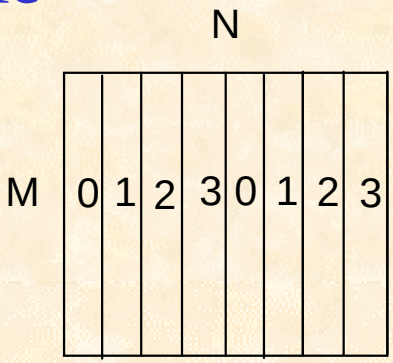
Meta Process Model

- Association of process and shared data
  - MpC shared data mapping API example



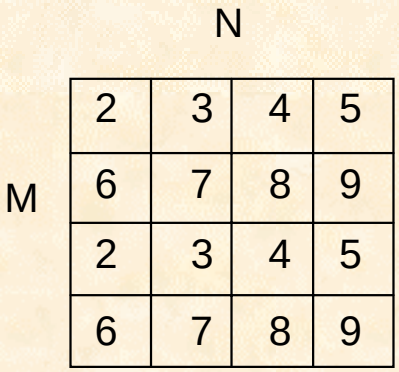
divide into 4, process num

shared int a[M][N]::[4][ ](0,4)



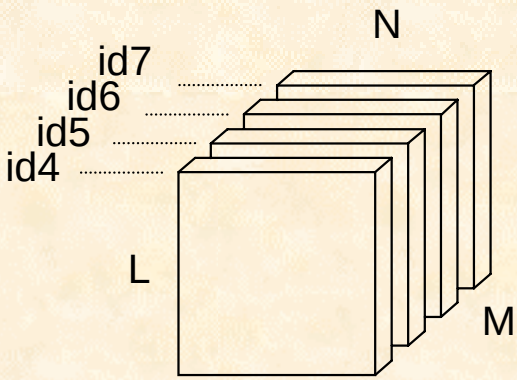
divide into N, column num

shared double a[M][N]::[ ][N](0,4)



divide into 8, process num

shared a[M][N]::[4][4](2,8)



divide into 4, process num

shared a[L][M][N]::[ ][4][ ](4,4)

Meta Process Model

- Relaxed memory consistency model for *shared* data

- Good Portability

  using universal API as in SDSM & pthread,
  lock and barrier

Implementation

1. For Cluster Computers
   uses user-level software DSM
   (TreadMarks, JIAJIA, SMS)
   which runs on various OSs and Architectures
   easy to install by general users

2. For Shared Memory Parallel Machines
   uses pthread library
   executable on various OS and Architectures
   for both NUMA and UMA machines

Meta Process Model

- Good Portability

  using universal API as in SDSM & pthread,
  lock and barrier

Implementation

1. For Cluster Computers
   uses user-level software DSM
   (TreadMarks, JIAJIA, SMS)
   which runs on various OSs and Architectures
   easy to install by general users

2. For Shared Memory Parallel Machines
   uses pthread library
   executable on various OS and Architectures
   for both NUMA and UMA machines

MpC Language (Meta Process C)

Language for realizing Meta Process Model

Minimum Invasion of ANSI C

- Easy to understand, to use and to port compiler

Only 2 Enhancements

- New data storage type: shared
- Distributed data mapping API

MpC Standard Function Library

Initiation	void mpc_init(int argc, char *argv)
Termination	void mpc_exit(int value)
Error Termination	void mpc_err(int value, char *msg)
Barrier	void mpc_barrier(int value)
Lock	void mpc_lock(int lockid)
Unlock	void mpc_unlock(int lockid)
Condition signal	void mpc_cond_signal(int condid)
Condition broadcast	void mpc_cond_broadcast(int condid)
Condition wait	void mpc_cond_wait(int condid, int lockid)
Share data allocation	void *mpc_alloc(size)
Share data allocation	void *mpc_alloc(char *declare)

MpC Constant

NPROCS: The number of processes consisting of a meta process

MYPID: 0 .. NPROCS-1 Process unique identifier number

MpC program example

```
#include <stdio.h>
#include <mpc.h>
#definr M 1024
#define N 2048

shared double matrix[M][N]::[NPROCS][ ](0,NPROCS);
shared double sum::(0);

int main(int argc, char **argv)
{
   FILE fp;
   double mysum=0;
   int start, end, i, j;

   mpc_init(argc, argv);
   if(MYPID == 0){
      fp=fopen("initial.dat", "r");
      for(i=0; i<M; i++)
         for(j=0; j<N; j++) fscanf(fp,"%f", &matirx[i][j]);
      sum = 0;
   }
   mpc_barrier(0);

   start = M/ NPROCS * MYPID;
    end = start+M/ NPROCS ;
    for(i=start; i<end; i++)
        for(j=0; j<N; j++)
            mysum += do_something(matrix[i][j]);

    mpc_lock(0);
      sum += mysum;
    mpc_unlock(0);

    mpc_barrier(0);
    if(MYPID  == 0)  printf("Result=%lf¥n",sum);
    mpc_exit(0);
}
```

MpC using SDSMs

on Cluster Computers

Uses SDSM for Distributed memory Systems

- Specify underlying execution system in compile parameter (SMS, TreadMarks, JIAJIA)
- MpC Standard function calls are translated to specified SDSM function calls
- Link with the specified SDSM library

Performance Evaluation
Hardware and Software Environment

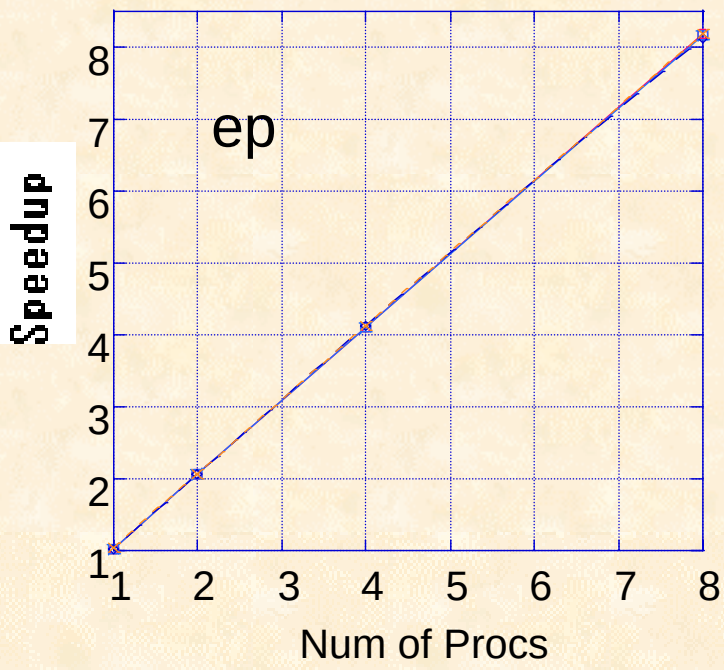
CPU	Intel PentiumIII-S 1.13GHz
Memory	512MB
Network	Intel PRO/1000T 3Com SuperStack3 Switch
OS	RedhatLinux7.1.2 kernel 2.4.7.10

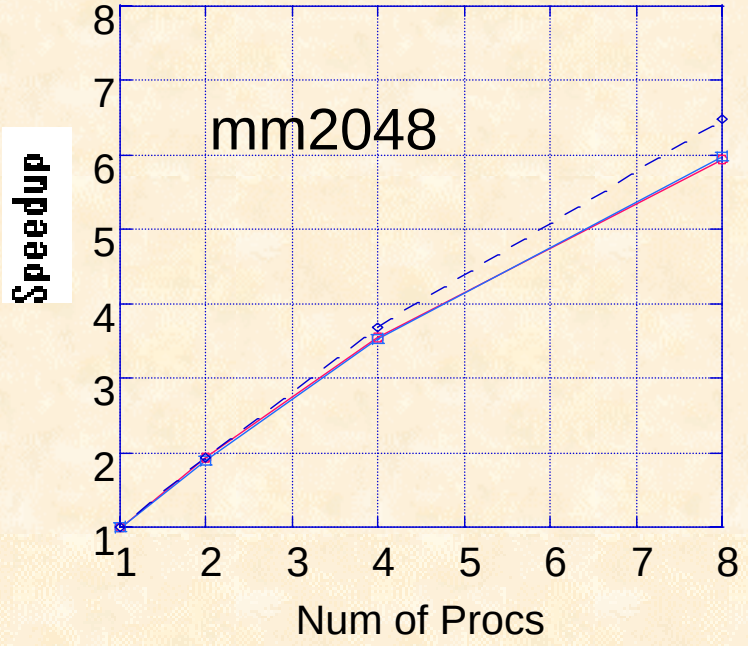
Compiler	gcc version 2.96 -O3
SDSM	SMS0.4.16
	TreadMarks 1.0.3.2
	JIAJIA 2.2

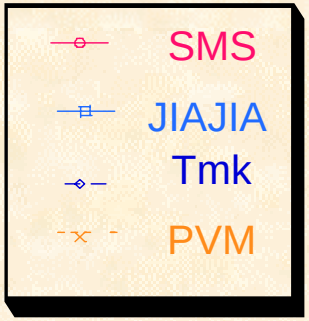
Benchmark Programs and their parameters

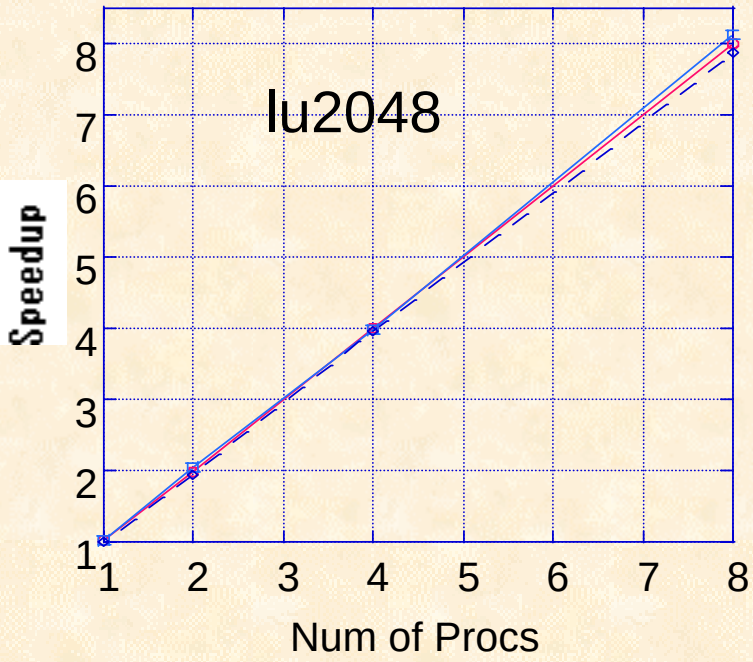
Progs	Parameters	Data Size	Barrier /proc	Lock /proc
ep	M=28,MK=10	44B	2	1
tsp	19cities(19b)	100MB	4	75-122
lu	2048 x 2048 double, 32bloks	34MB	135	0
mm	2048 x 2048 double	96MB	3	0

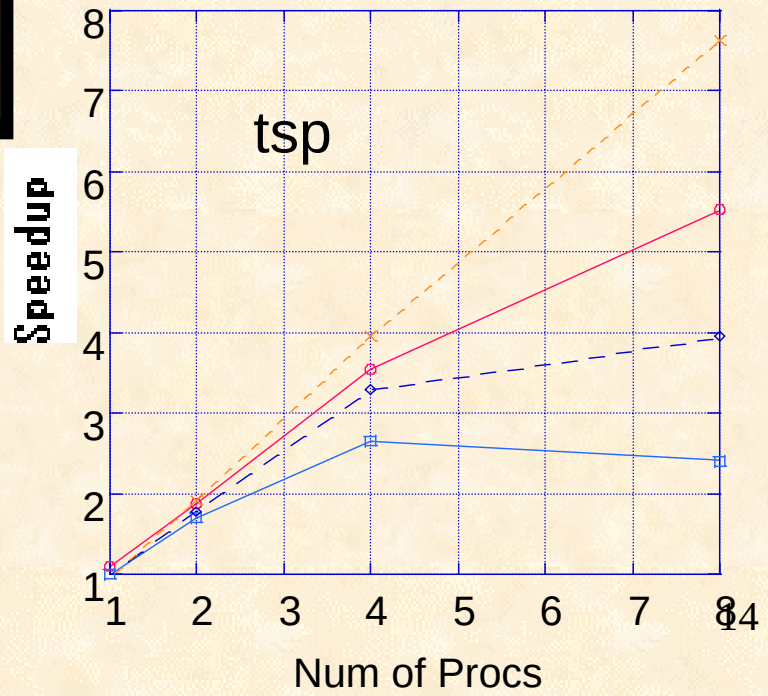
MpC Performance on various SDSMs











MpC Program for Cluster Computers

- Executable on various SDSMs with No Modification

- Executable on a wide variety of Architectures and OSs

  TreadMarks :AIX(SR6000), Linux(Alpha/x86), HPUX, SunOS/Solaris(SPARC/x86), IRIX(SGI), FreeBSD

  JIAJIA: SunOS/Solaris(SPARC),AIX(SP2),Linux(x86)

  SMS: Linux,FreeBSD(X86)

- No overhead between MpC program execution and direct SDSM program execution

MpC vs. OpenMP on Cluster Computers

Execution Environment

	MpC	OpenMP（Omni)
CPU	Intel PentiumIII 1.13GHz	
Num of CPU	Single CPU /nore, 1-8nodes	
memory	512MB	
OS	Red Hat Linux 7.2 (SCore5.6.1)	
kernel	2.4.21-1SCORE	
network	Giga Ethernet	Gigathernet, Myrinet
protcol	socket(UDP)	PM
SDSM	SMS	SCASH

MpC vs. OpenMP on Cluster Computers

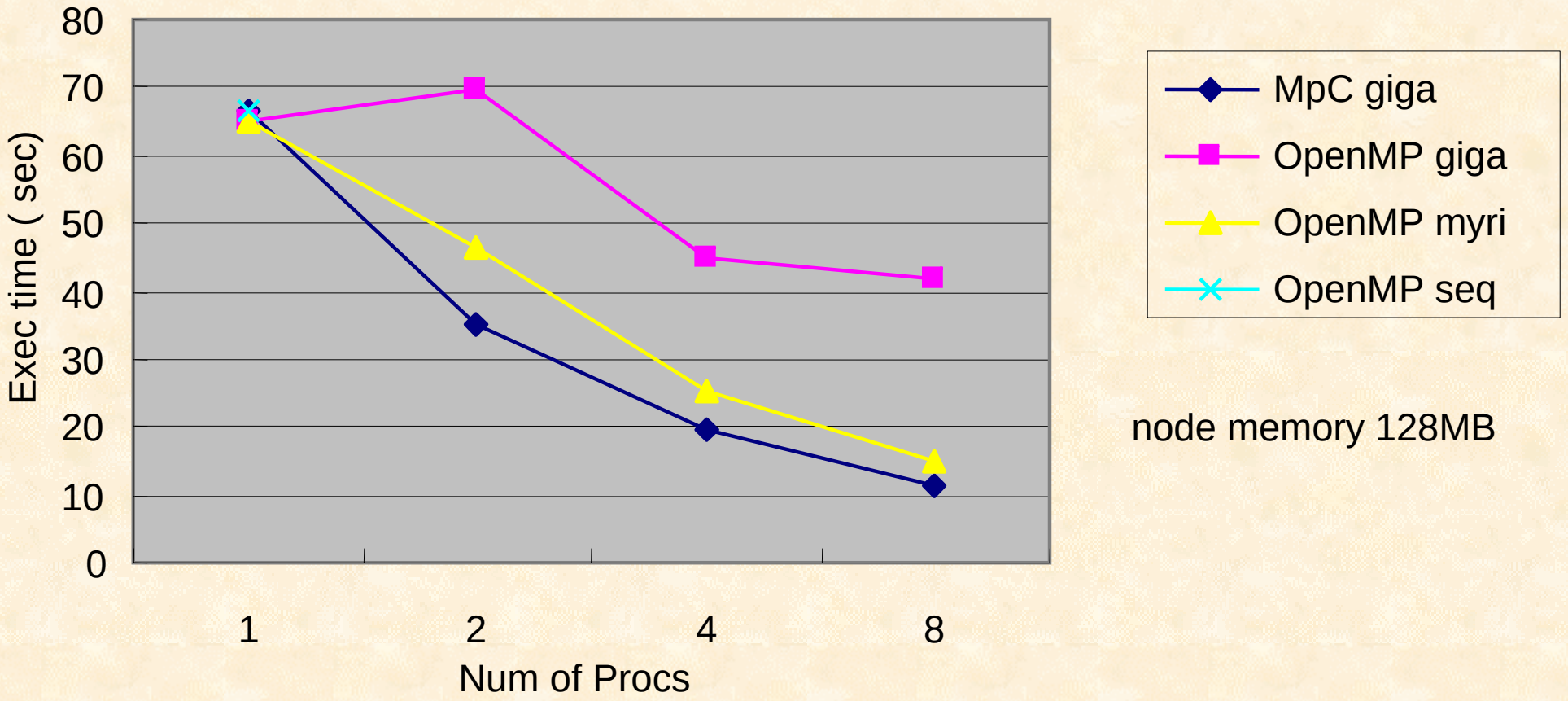
Compile Environment

	MpC	OpenMP(Omni)
compiler	mpcc	omcc 1.6
gcc version	2.96	
optimization	-O3	
OS	Red Hat Linux 7.3	
kernel	2.4.18	

MpC vs. OpenMP on Cluster Computers

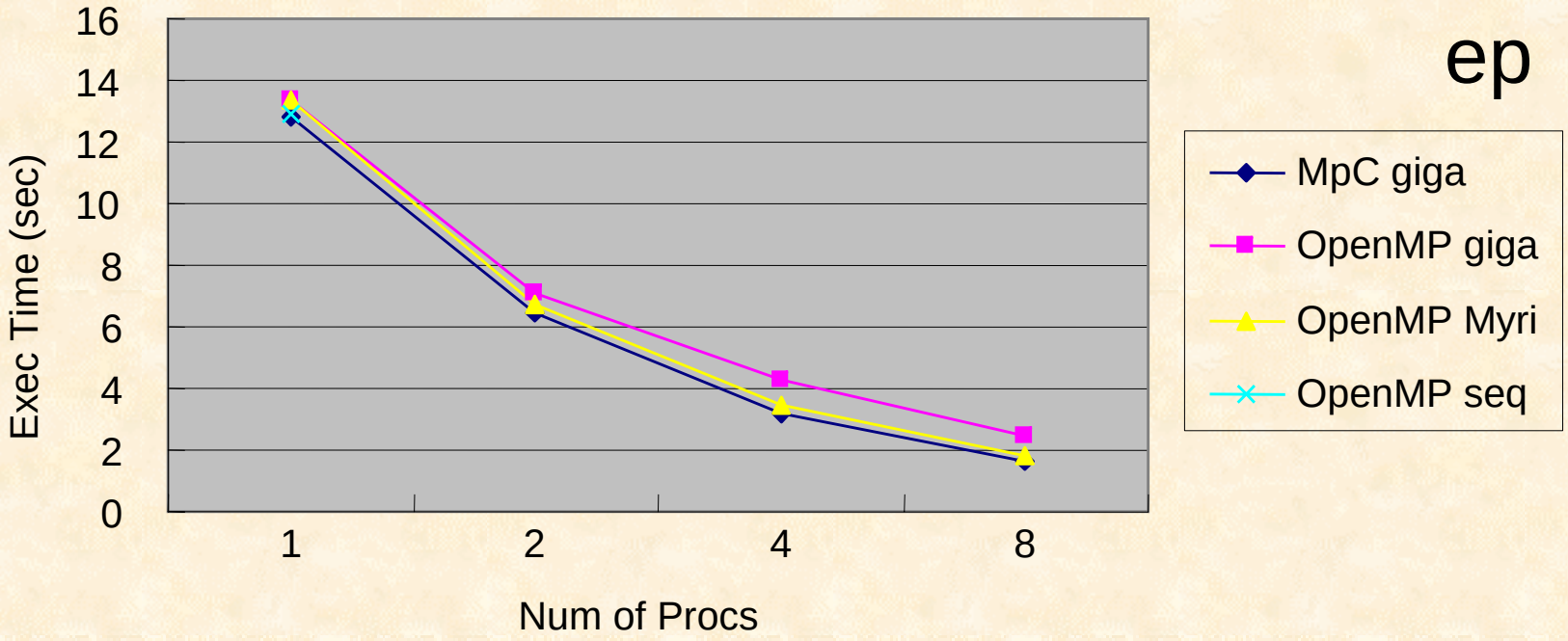
Benchmark programs

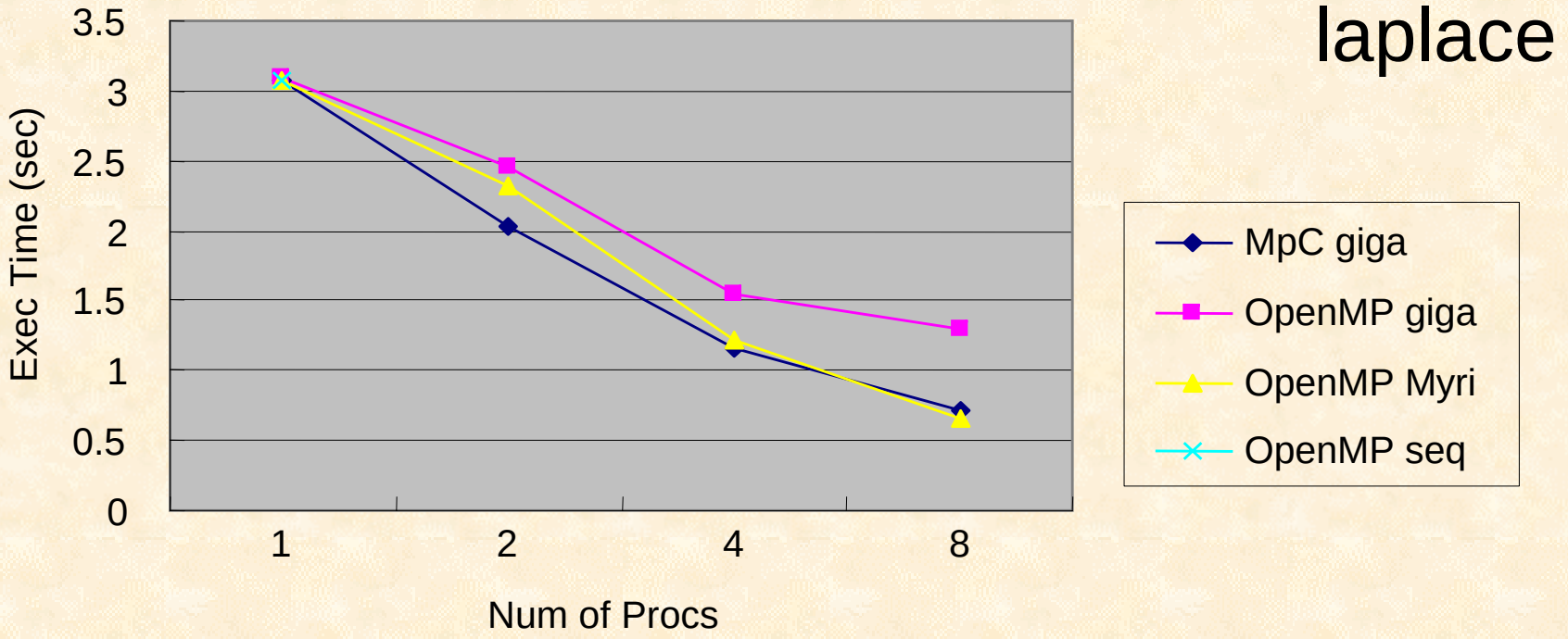
Programs	parameters	shared data size	shared mapping scheme
floyd	1024 x 1024 double	12MB	horizontal band
laplace	1024 x 1024 double	16MB	horizontal band
mandel	1024 x 1024 char	1MB	allocated in proc0
mm	1024 x 1024 double	24MB	horizontal band
galaxy	1000 bodies 10steps 100time	72KB	allocated in proc0
ep	S	80B	allocated in proc0

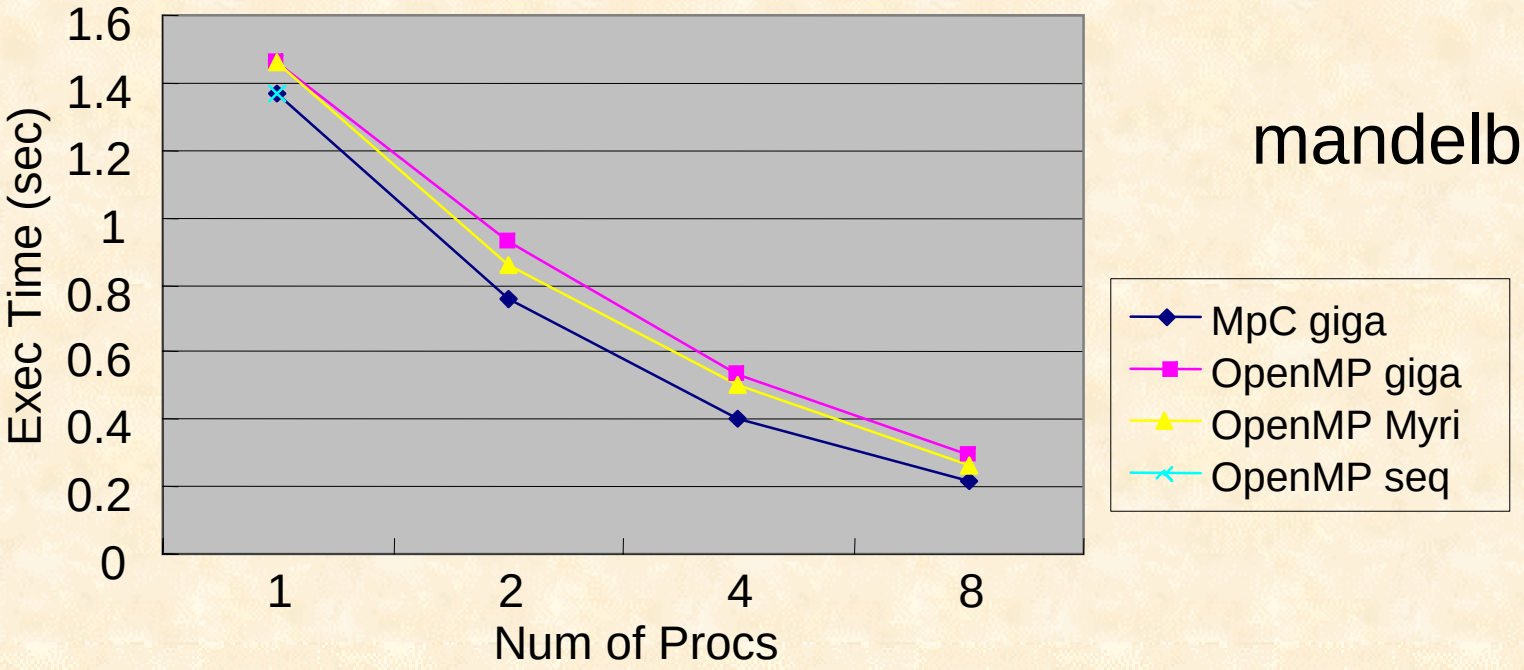


node memory 128MB

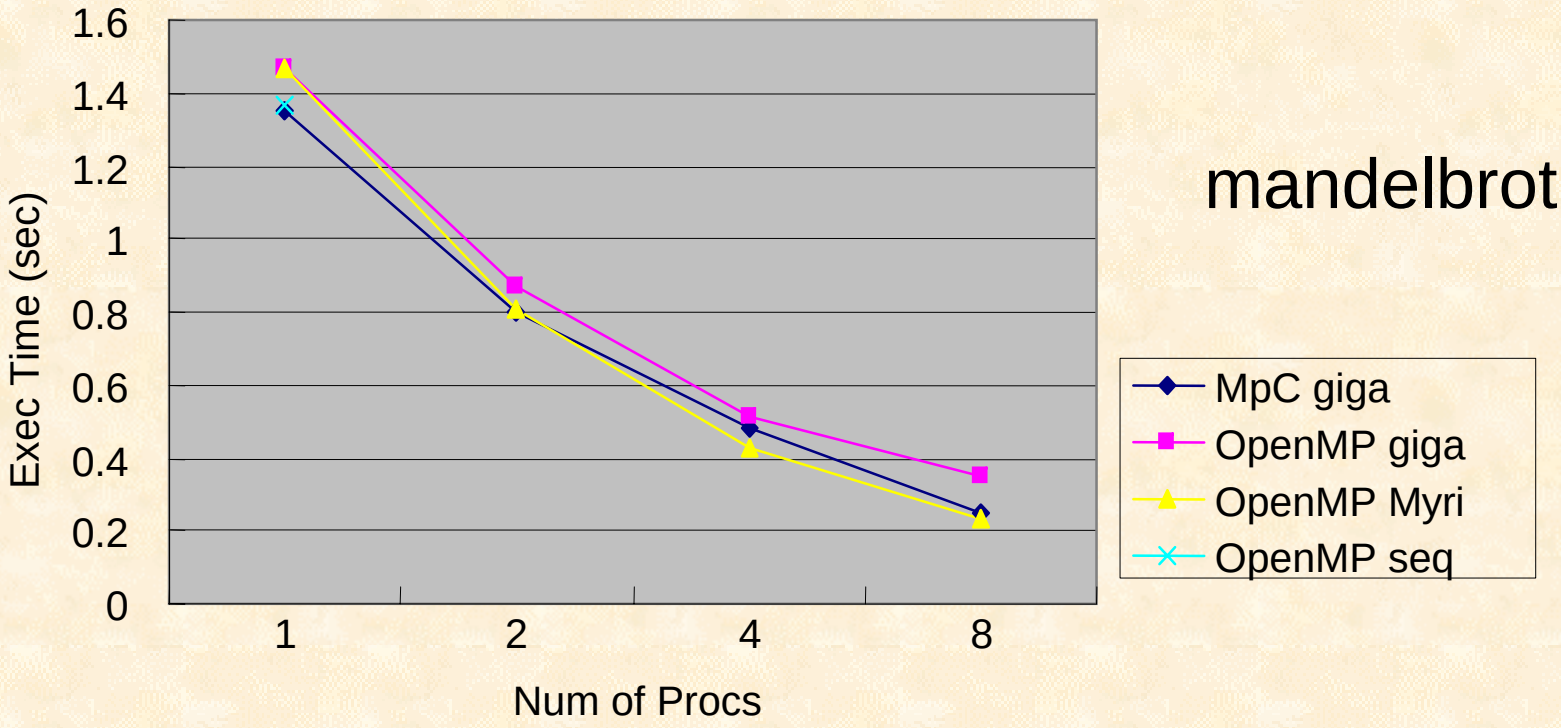
floyd : Shortest path search


ep
Exec Time (sec)
16
14
12
10
8
6
4
2
0
1
2
4
8
Num of Procs
MpC giga
OpenMP giga
OpenMP Myri
OpenMP seq

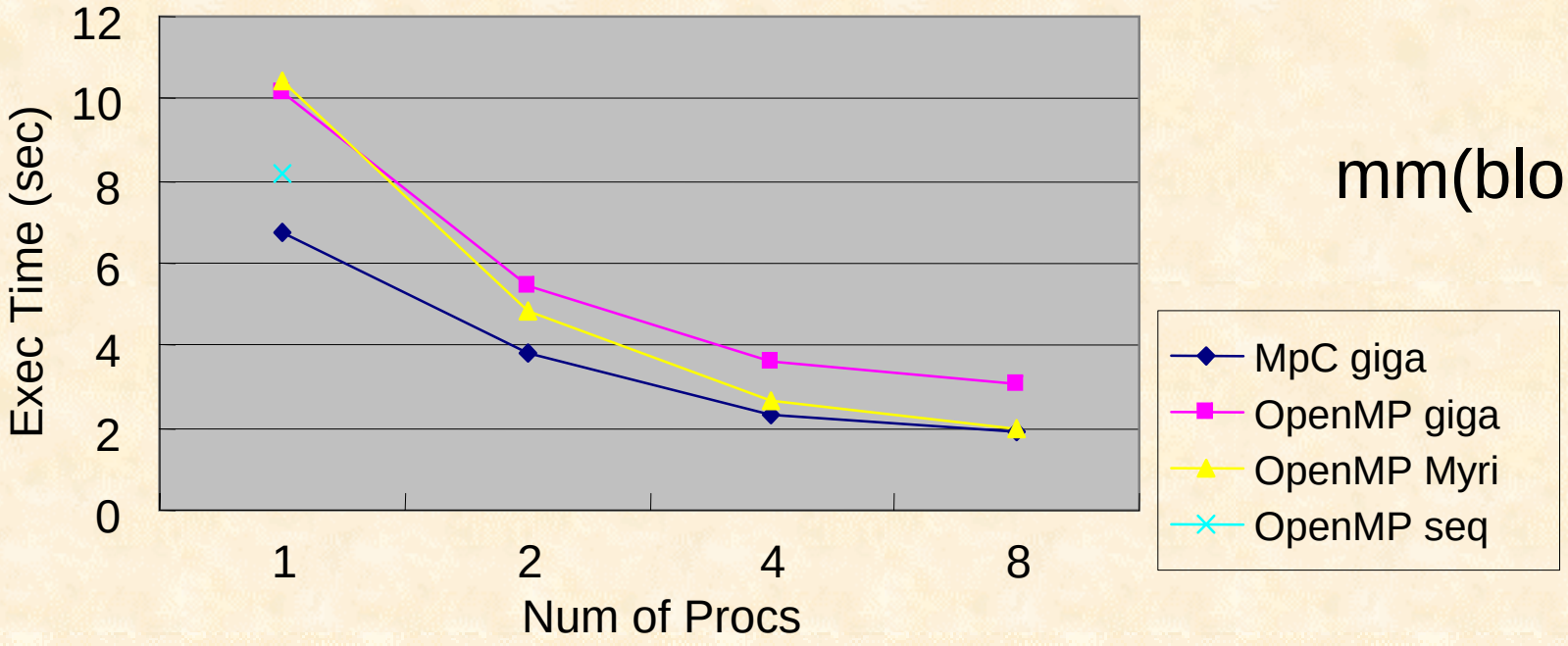

laplace
Exec Time (sec)
3.5
3
2.5
2
1.5
1
0.5
0
1
2
4
8
Num of Procs
MpC giga
OpenMP giga
OpenMP Myri
OpenMP seq



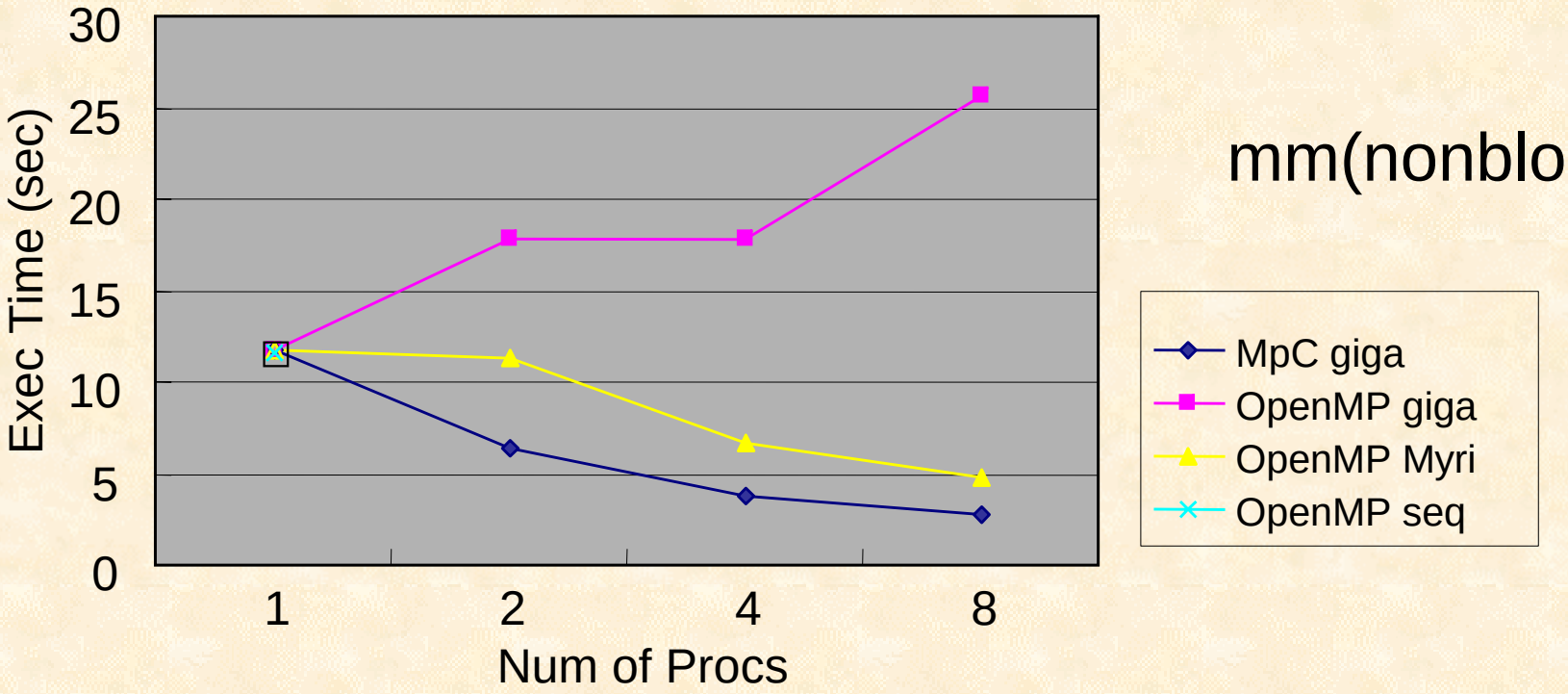
mandelbrot(static)



mandelbrot(dynamic)



mm(blocking)



mm(nonblocking)

MpC vs. OpenMP on Clusters Summary

Programs	Shared Data Size	Shared Data Access Conflict Level	Result Better one Giga/Myri
floyd	12MB (M)	High	MpC
ep	80B(S)	No	MpC/Comparable
laplace	16MB (L)	Little	MpC/Comparable
mandel (static)	1MB(S)	No	MpC
mandel (dynamic)	1MB(S)	No	MpC/Comparable
mm (blocked)	24MB(L)	High	MpC
mm (nonblocked)	24MB(L)	High	MpC
galaxy	72KB (S)	Little	Unreliable(OpenMP)

MpC vs. OpenMP on Clusters

- MpC with Ethernet has
  Better Performance to OpenMP with Ethernet

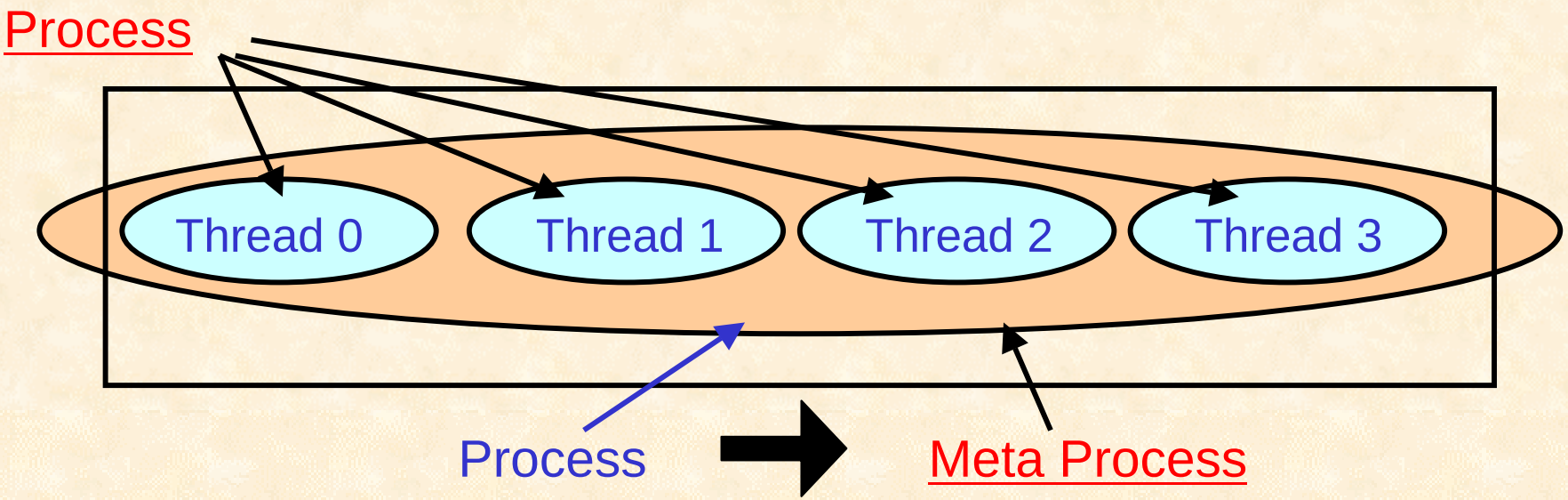
- MpC with Ethernet has
  Comparable Performance to OpenMP with Myrinet

MpC has Better performance than OpenMP
in the same Hardware Environment

MpC using pthread

on Share Memory Parallel Machines

Meta Process Model Execution mapping for Shared Memory Parallel Machine

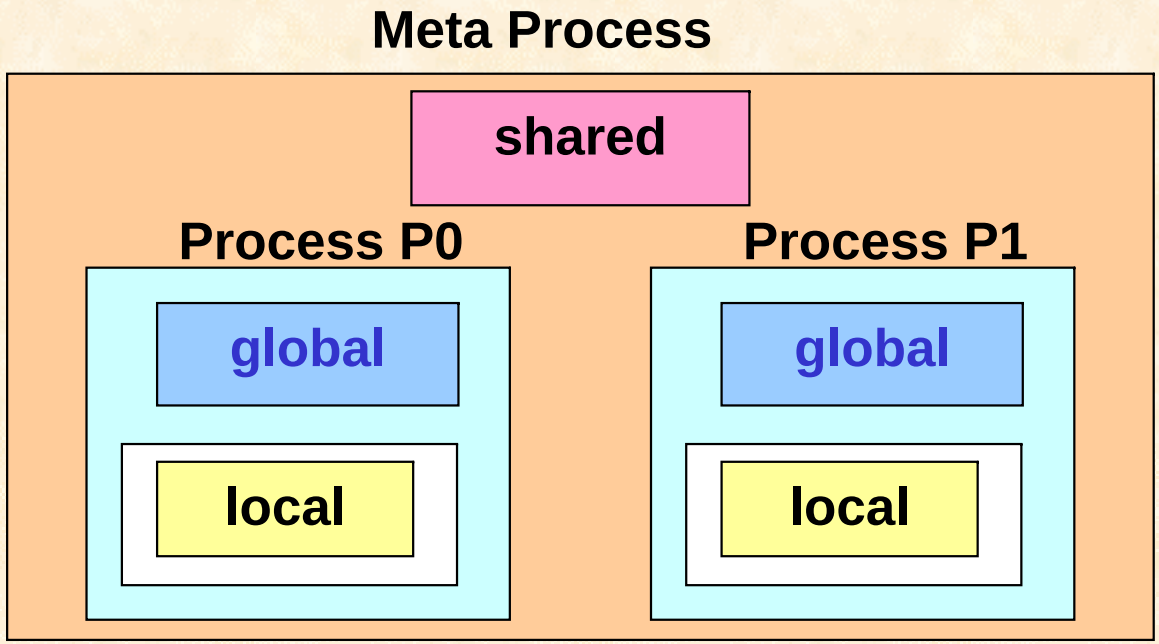


Ordinary OS Term

Meta Process Model Term

	SDSM, Cluster Implementation	pthread, Shared Memory Machine Implementation
One Application (Meta Process)	1 Group of Processes	1 Process
Parallel execution entity	1 Process	1 Thread

Meta Process Model Data mapping for Shared Memory Parallel Machine



Data Type in Meta Process Model	SDSM, Cluster Implementation	pthread, Shared memory Machine Implementation
shared	Process Shared (uses SDSM)	Thread Shared (global data)
global	Process Local (global data)	Thread Local (local data)
local	Process Local (local data)	Thread Local (local data)

Shared Memory parallel machine

Benchmark Result (1) Execution Time

(sec)

programs	Num of Proc	MpC pthread smp 2CPUs LINUX 2.4.20-6smp RedHat9	MpC pthread rs6000 4CPUs AIX5.2	MpC SMS Gigabit Ether pc cluster LINUX 2.4.20 RedHat9
ep S class	1	10.82	10.45	11.19
	2	5.54	5.23	6.09
	4		2.65	2.56
	8			1.28
mandeld 1024x1024 0.3<x<0.4 0.5<y<0.6	1	1.35	1.03	1.39
	2	0.68	0.53	0.87
	4		0.38	0.54
	8			0.43

Shared Memory parallel machine

Benchmark Result (2) Execution Time (sec)

programs	Num of Proc	MpC pthread smp 2CPUs LINUX 2.4.20-6smp RedHat9	MpC pthread rs6000 4CPUs AIX5.2	MpC SMS GigabitEther pc cluster LINUX 2.4.20 RedHat9
galaxy 1000body 10 steps 100time	1	9.02	8.81	8.17
	2	4.52	4.42	4.53
	4		2.24	2.53
	8			1.63
mm512 512x512 double array blocked	1	0.77	0.61	0.80
	2	0.40	0.31	0.47
	4		0.17	0.31
	8			0.29
mm1024 1024x1024 double array blocked	1	6.18	4.9	6.90
	2	3.17	2.47	3.47
	4		1.49	2.04
	8			1.57

MpC vs. OpenMP on Shared Memory Machines

Comparable Performance on the same SMP

(sec)

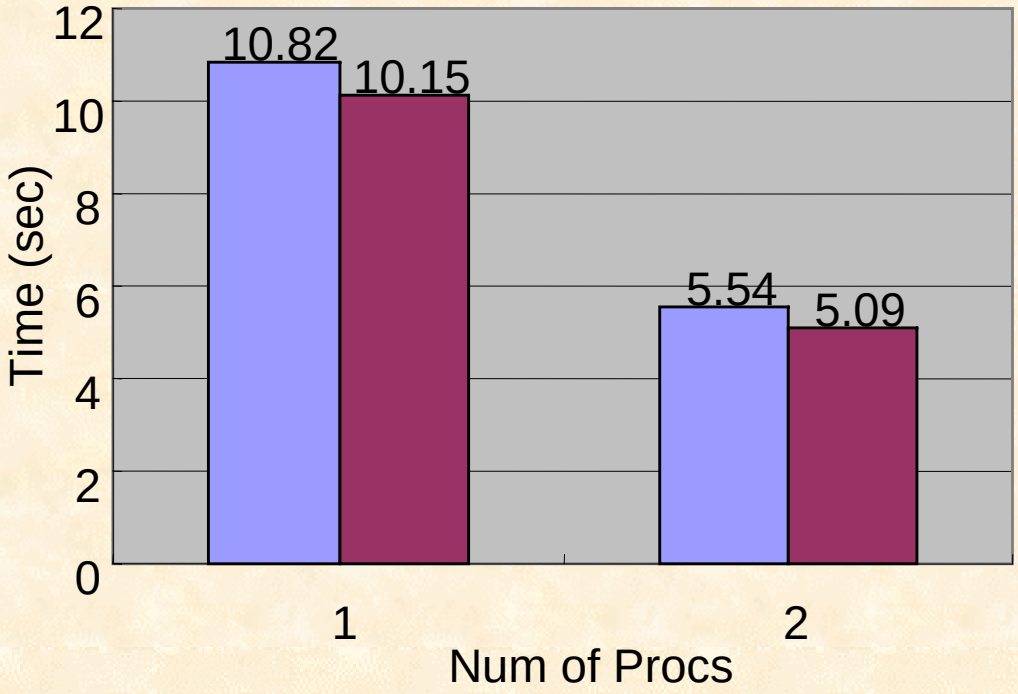
Language		MpC	OpenMP	C
Compiler		mpcc	omcc	gcc
floyd	1	81.35	100.68	85.23
	2	60.8	67.66	
laplace	1	3.7	4.12	3.74
	2	3.02	2.96	
mandelbrot	1	2.21	2.41	2.2
static	2	1.25	1.37	
mandelbrot	1	2.21	2.16	2.2
dynamic	2	1.11	1.09	
mm	1	13.72	13.23	13.54
blocking	2	6.95	26.24	
mm	1	23.52	23.74	23.48
nonblocking	2	17.64	17.47	
galaxy	1	15.36	14.99	14.18
	2	7.71	(4.1)	
ep	1	23.94	27.13	29.11
	2	12.3	16.72	

SMP machine 2CPU Pentium3 700MHz, 256MB,Linux 2.4.21-1SCORE

MpC vs. UPC on Share Memory Parallel Machines

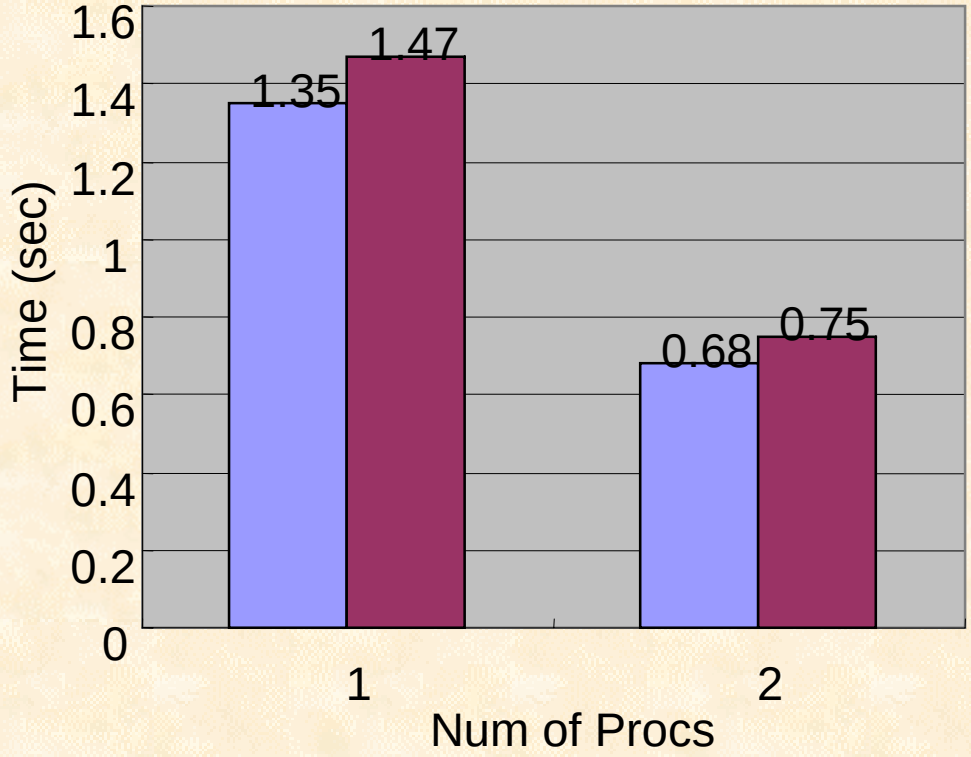
gnuUPC for x86 SMP

2CPU SMP Pentium3 1.13GHz
Linux2.4.20-6smp



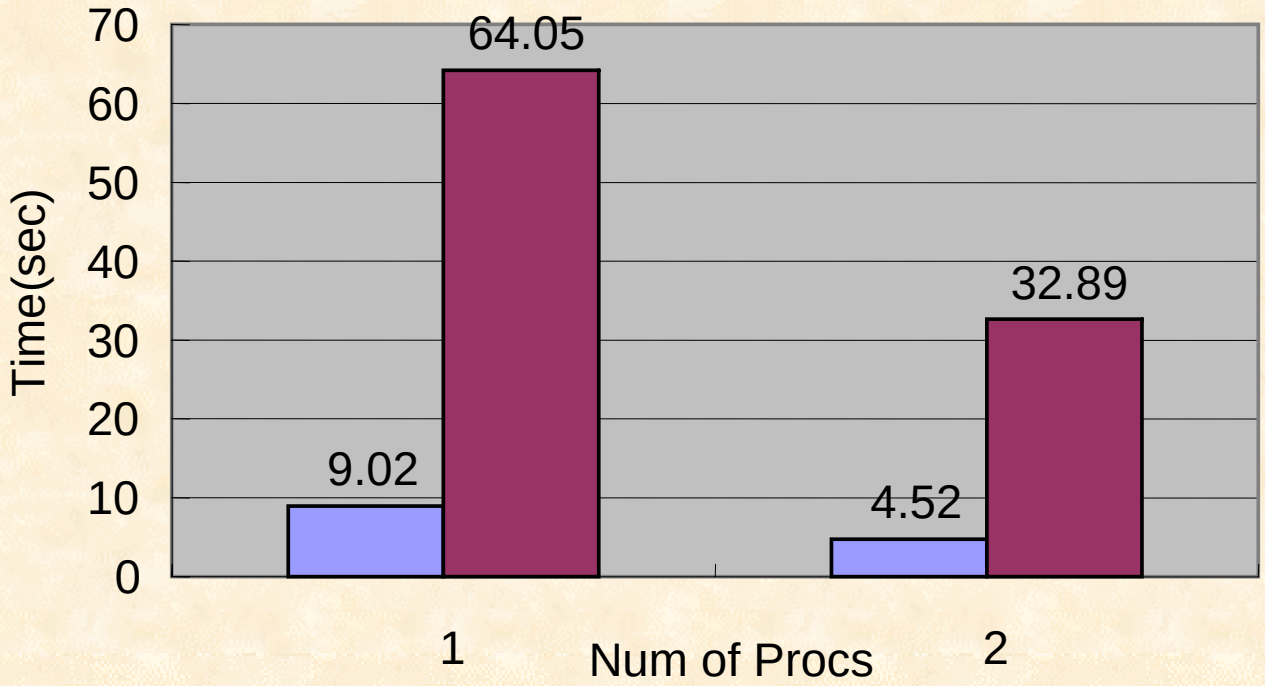
ep

MpC 0proc
gnuUPC 1element



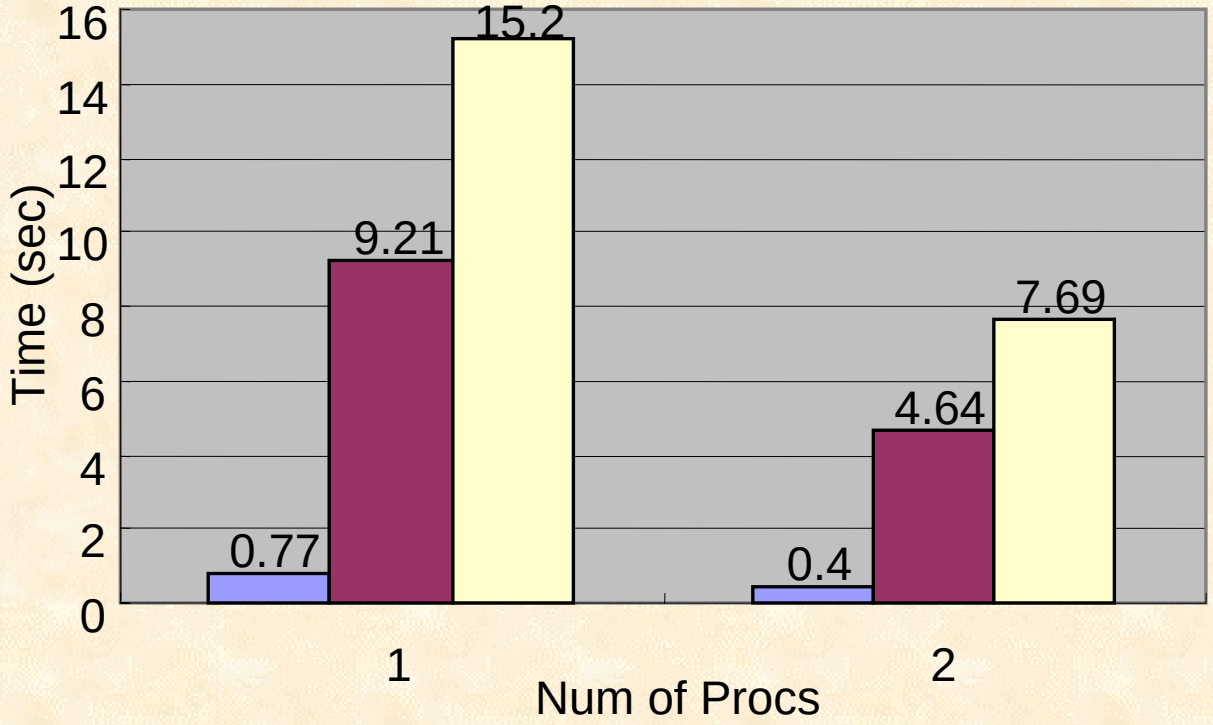
mandel dynamic

MpC 0proc
gnuUPC 0proc



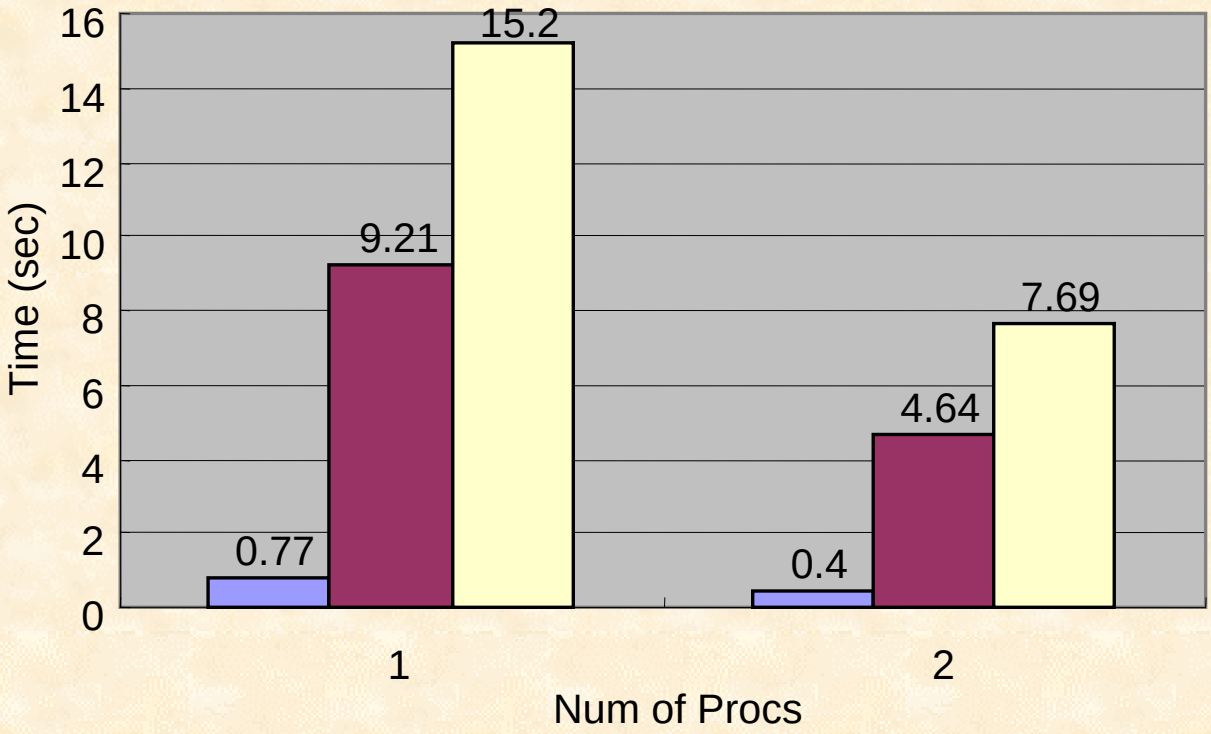
galaxy (n-body problem)

MpC
gnuUPC

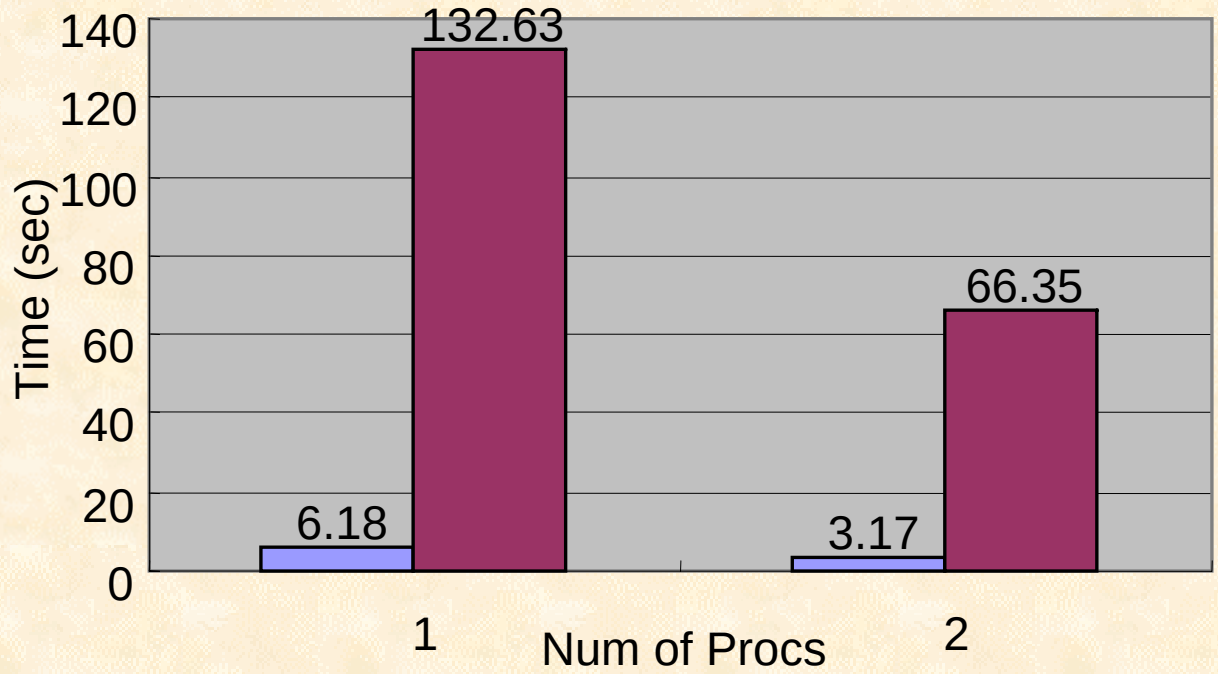


mm512 (matrix mul)

MpC
gnuUPC 0proc
gnuUPC horizontal



mm512 (matrix mul)



mm1024 (matrix mul)

MpC vs. Other Languages on Shared Memory Machines

- OpenMP(Omni)
  - MpC has comparable performance to OpenMP
- UPC(gnuUPC)
  - MpC is comparable in some programs(ep,mandel)to UPC
  - MpC is much faster in other programs with high memory access than UPC

MpC program performance is good

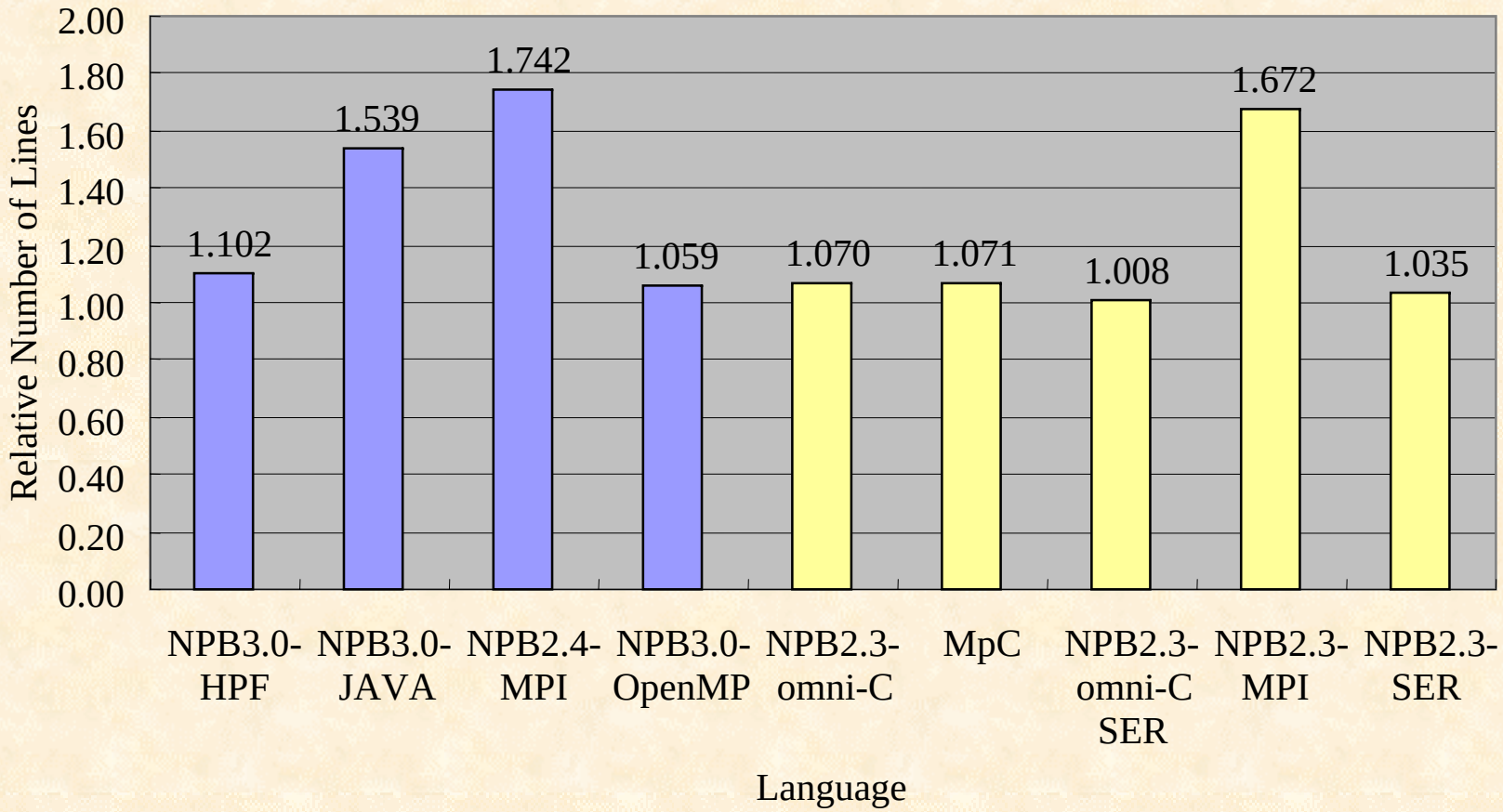
both on clusters and shared memory machines

MpC Programming Productivity

One of the indices of Programming Productivity Effective Line counts of Benchmark programs

	Number of Effective Lines			Relative Value to OpenMP	
Programs	MpC	UPC	OpenMP	MpC	UPC
floyd			71	1.141	
laplace			66	1.258	
mandeld	138	127	106	1.302	1.198
mandels	112		98	1.143	
mm bloked	104	100	91	1.143	1.099
mm nonblocked			82	1.11	1.11
galaxy	191	200	180	1.061	1.111
ep	174	172	154	1.13	1.117
Average				1.161	1.127

One of the indices of Programming Productivity
Average of Relative Line counts
to NPB3.0-SER programs



Summary

- Meta Process Model

  New Distributed Shared Memory Model
- MpC Programs
  - Executable on various OSs and Architectures, cluster computers and shared memory machines
  - MpC Compiler Highly portable
  - No message passing statements, good readability
  - Explicit data distribution and synchronization
  - Performance:

    Comparable to or better than OpenMP, UPC
- MpC : Another alternative for parallel programming API to OpenMP and MPI

Thank you !