Laboratorio di Calcolo Parallelo

Lezione 3: Aspetti pratici

Francesco Versaci & Alberto Bertoldo

Università di Padova

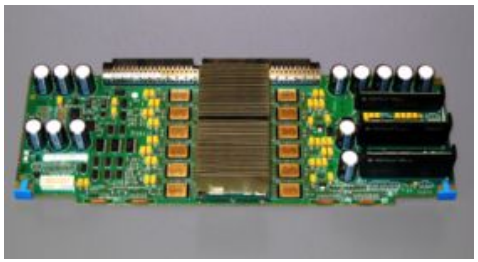
19 maggio 2009

DEPARTMENT OF INFORMATION ENGINEERING

UNIVERSITY OF PADOVA

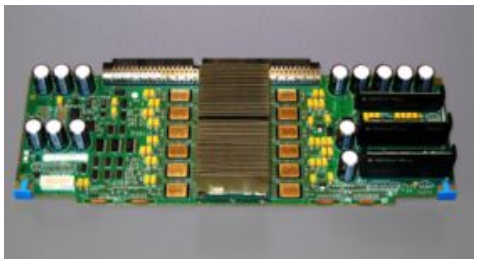
Sommario

Sommario



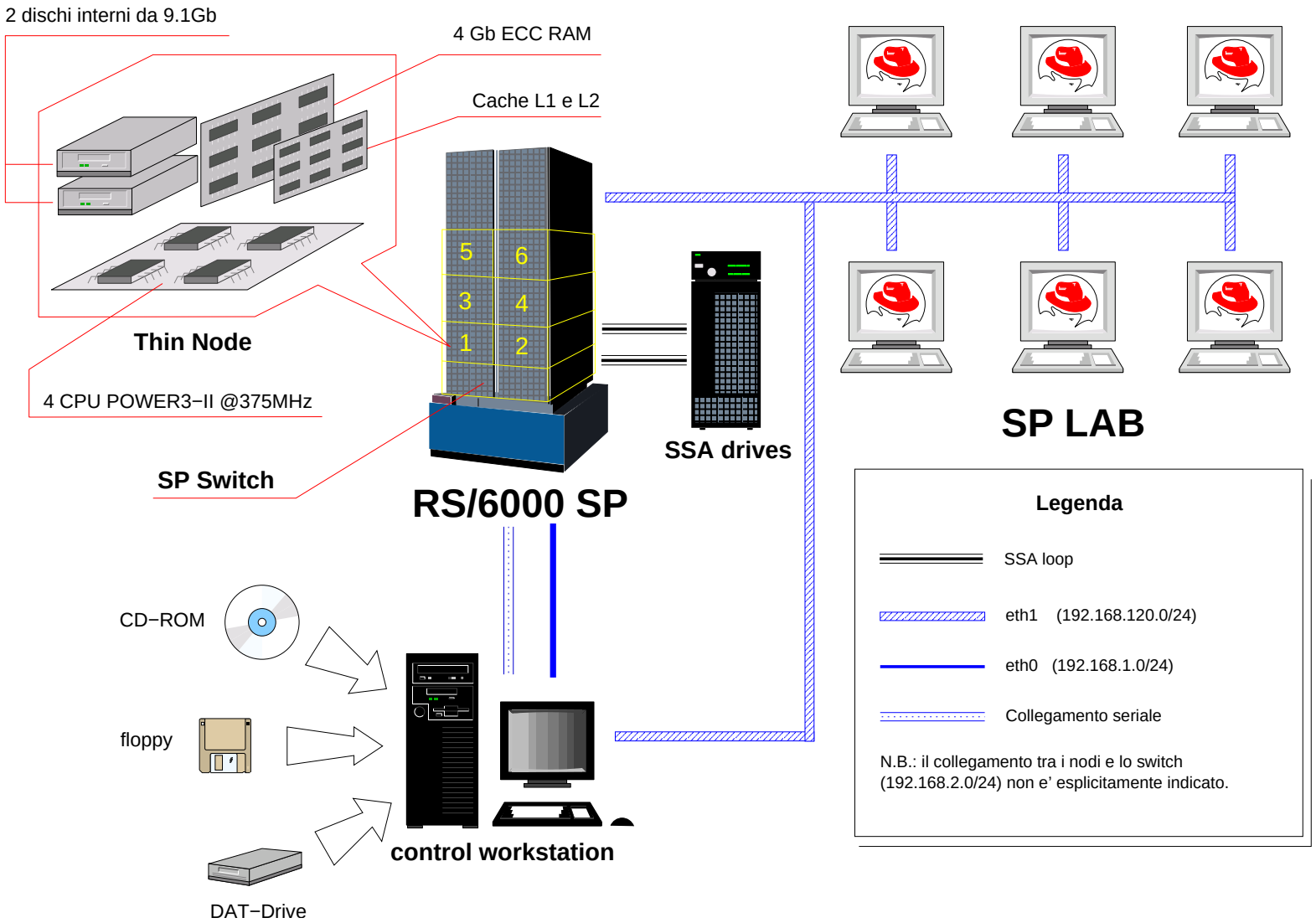
Caratteristiche:

Model	IBM SP RS/6000 Power3
Processor (PE)	Power3-II @ 375 MHz
Number of PEs	24
Processor per node	6 nodes with 4 proc.
DRAM	24 Gbytes (4 GB/node)
Disk space	510 GB
Peak performance	36 Gflop/s
OS	AIX 4.3.3
Internal Network	SP Switch MX2
Available compilers	Fortran F90, C, C++
Parallel libraries	MPI



Architettura di memoria:

- Cluster di SMP
- Ogni nodo è un SMP
  - Possiede un nome: `spnd0x` con $\mathtt{x} \in \{1, \ldots 6\}$
  - Processori uguali **indistinguibili**
  - Memoria RAM condivisa
  - Accesso uniforme (UMA)
- I nodi sono collegati tramite una rete dedicata
  - Rete Ω doppia
  - Memoria distribuita
  - Accesso a memoria remota non possibile


2 dischi interni da 9.1Gb
4 Gb ECC RAM
Cache L1 e L2
Thin Node
4 CPU POWER3−II @375MHz
SP Switch
5
6
3
4
1
2
RS/6000 SP
SSA drives
SP LAB
CD−ROM
floppy
DAT−Drive
control workstation
Legenda
SSA loop
eth1 (192.168.120.0/24)
eth0 (192.168.1.0/24)
Collegamento seriale
N.B.: il collegamento tra i nodi e lo switch
(192.168.2.0/24) non e' esplicitamente indicato.

Accesso remoto

Login

- L'accesso avviene unicamente via `ssh`
- Dipende **da dove** vi state collegando (rispetto alla rete DEI):

  interno: `ssh user@192.168.120.201`
  esterno: `ssh user@splab.dei.unipd.it -p9001`
- Vi collegate a `spnd01`

Copia di file

- La copia avviene unicamente via `scp`
- Dipende **da dove** state lanciando il comando:

splab: `scp` $file$ `miouser@miohost:`$path$

esterno: `scp -P 9001` $file$ `user@splab.dei.unipd.it:`$path$

NOTE

- Per avere un account mandatemi al piú presto un'email
- L'accesso dall'esterno passa per il firewall: `9001` è la porta

Esercizio 1
Accesso remoto

1. Aprire un client SSH
2. Collegarsi al nodo `spnd01` con username e password
3. Copiare il file `/ext/calcolo-parallelo/code/hello.c` sulla propria home
4. Aprire il file locale con un editor (`vi` o `emacs`)

Sommario

Compilazione

- Compilatori MPI = `mp` + compilatore
- Sono script: richiamano i compilatori standard IBM + link alle librerie MPI

C: `mpcc, mpcc_r`

C++: `mpCC, mpCC_r`

Fortran: `mpxlf, mpxlf90, mpxlf95` (e le versioni `_r`)

Opzioni comuni

Standard: `-c`, `-o` *nome*, `-I`*dir* `-L`*dir* `-l`*lib*

Ottimizzazione: `-O`*n* `-qarch=pwr3 -qtune=pwr3 -qcache=auto`

Debug: `-g` e `-C` (Fortran)

Modalità di esecuzione

Interattiva: Esecuzione immediata, per il debug, non adatta all'analisi delle prestazioni

Batch: Utilizza un sistema di gestione basato su code

Interattiva

- Direttamente tramite il *Parallel Operating Environment*
- Es: `poe ./a.out -procs 4 -labelio yes`
- oppure: `./a.out -procs 4 -labelio yes`
- Richiede un file che descrive la macchina: `host.list`

Opzioni

- Opzioni di `poe`
- Variabili di ambiente
- Direttive a LoadLeveler

Esercizio 2
Programmare

1. Compilare `hello.c` con `mpcc hello.c -o hello`
2. Provare a eseguire `./hello`
3. Copiare il file `/ext/calcolo-parallelo/host.list` da SP al Desktop locale
4. Aprire il file locale con un editor
5. Copiare il file su SP nella propria home
6. Riprovare a eseguire `./hello`
7. Provare a eseguire `./hello -procs 4`
8. Provare a eseguire `poe -h`
9. Provare a eseguire `./hello -procs 4 -labelio yes`
10. Provare a eseguire `./hello -procs 4 -stdoutmode 0`
11. Provare a eseguire `./hello -procs 4 -stdoutmode ordered`

Sommario

Cos'è?

È un sistema per la gestione dei **job**
job = esecuzione di un programma (seriale o MPI) su un sistema IBM

Perché si usa?

- Permette una migliore allocazione delle risorse di sistema
- Ottimizza l'esecuzione
- Suddivide il carico tra i processori
- Permette un utilizzo equo da parte di più utenti
- Permette l'esecuzione batch (non interattiva)

Quando si deve usare?

- Il più possibile, perché aumenta l'efficienza del sistema
- Non è adatto per fare debug

Esempio di job file

```
#! /bin/bash

#@ job_name         =  helloworld
#@ initialdir       =  /home/versacif/hello
#@ input            =  /dev/null
#@ output           =  $(job_name).out
#@ error            =  $(job_name).err
#@ class            =  short
#@ job_type         =  parallel
#@ node_usage       =  shared
#@ blocking         =  unlimited
#@ total_tasks      =  8

#@ environment      =  MP_SHARED_MEMORY=yes; MP_LABELIO=yes
#@ network.mpi      =  switch,shared,US

#@ queue
./hello
```

http://www.dei.unipd.it/~addetto/manuali_online/SP/LLUAdmin/lllv2mst85.html

Permettono di interagire con LoadLeveler

Comandi principali

Nome	Descrizione
`llsubmit`	Sottomettere un job file per l'esecuzione di un programma
`llq`	Controllare lo stato di un job
`llcancel`	Cancellare un job precedentemente sottomesso
`llstatus`	Controllare lo stato della macchina
`llclass`	Ottenere la lista delle code di esecuzione

`http://www.dei.unipd.it/~addetto/manuali_online/SP/LLUAdmin/lllv2mst200.html`

Esercizio 4
Job command file

1. Copiare il file `/ext/calcolo-parallelo/jobfile/hello.job` da SP al Desktop locale
2. Aprire il file locale con un editor e sistemarlo
3. Copiare il file su SP nella propria home
4. Provare a sottomettere il job usando i comandi di `LoadLeveler`
   - `llclass`, `llstatus`, `llsubmit`, `llq`, `llcancel`

NOTE

- Ricordatevi di cancellare i job bloccati!!!

Sommario

Misure dirette

- Tempo di esecuzione aggregato $T(n)$ con n processi
- Numero di istruzioni eseguite (in genere floating-point)
  - Ricavato analiticamente
  - Ricavato sperimentalmente (vedi HPM Toolkit)
- Sfruttamento delle risorse di calcolo (vedi HPM Toolkit)
  - ES: L1 miss, TLB miss, FMA

Misure indirette

- Scalabilità: $S(n) = \frac{T(1)}{T(n)}$
- flop/s: $F(n) = \frac{flop}{T(n)}$
- Efficienza di calcolo: $E_c(n) = \frac{F(n)}{peak(n)} = \frac{F(n)}{n \cdot 1.5 \cdot 10^9}$

- Differenti algoritmi
- Differenti parametri dell'algoritmo
  - taglia dei blocchi
  - layout dei dati
  - ...
- Differenti protocolli di comunicazione
  - Primitive bloccanti o non bloccanti
  - Modalità diverse
  - Primitive collettive
  - ...
- Numero di processi utilizzati (da 1 a 24)
- Taglia dell'input
- Differenti parametri di esecuzione
  - Mappa processi/processori
  - Rete di comunicazione: switch o ethernet
  - Uso della memoria condivisa: MP_SHARED_MEMORY (yes o no)
  - Buffering intermedio: MP_EAGER_LIMIT (default = 4K, max = 64K)

Cos'è?

È una libreria scritta da IBM per accedere ai **contatori hardware**

Alcuni contatori del Power3

- Cicli macchina
- TLB misses
- Cache misses
- Floating point operations
- Load e Store
- ... e molti altri

`http://www.hpcx.ac.uk/support/documentation/IBMdocuments/HPM.html`

```
#include "libhpm.h"
...
hpmInit( taskID, "my program" );
hpmStart( 1, "outer call" );
do_work();
hpmStart( 2, "computing meaning of life" );
do_more_work();
hpmStop( 2 );
hpmStop( 1 );
hpmTerminate( taskID );
...
```

NOTE

- In compilazione va aggiunto: `-lhpm -lpmapi -lm`
- Quando termina crea dei file con i risultati

Esempi

Prove di esecuzione

1. Copiare tutti i file da `/ext/calcolo-parallelo/code/esempi`
2. Aprire i file locali con un editor e cercare di capire cosa fanno
3. Copiare il file su SP nella propria home e provarli in interattivo

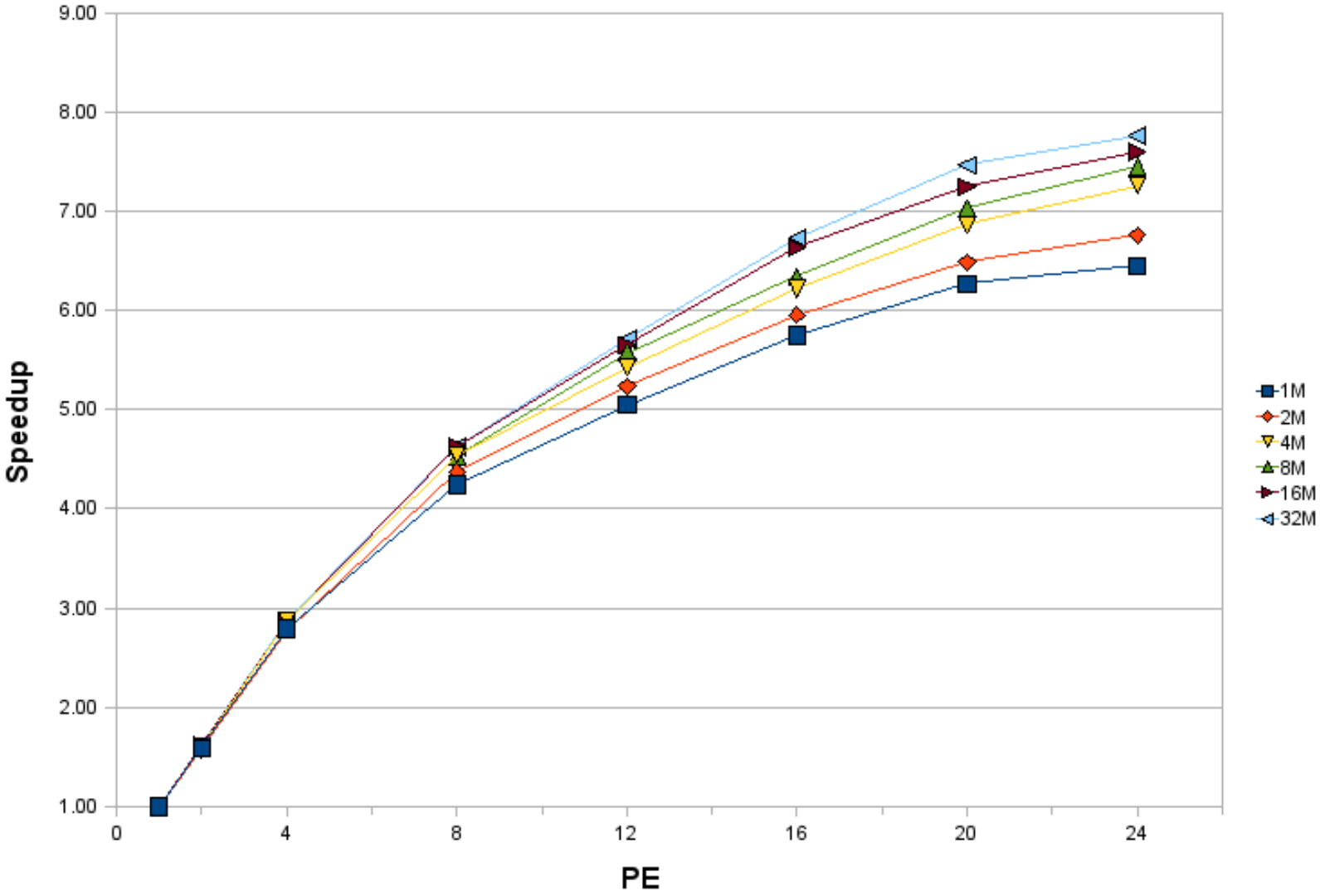
Misurare le prestazioni

1. Scegliete un programma della prima prova
2. Aggiungete le istruzioni per misurarne le prestazioni con `MPI_Wtime` e `HPM Toolkit`
3. Provare e analizzare i risultati

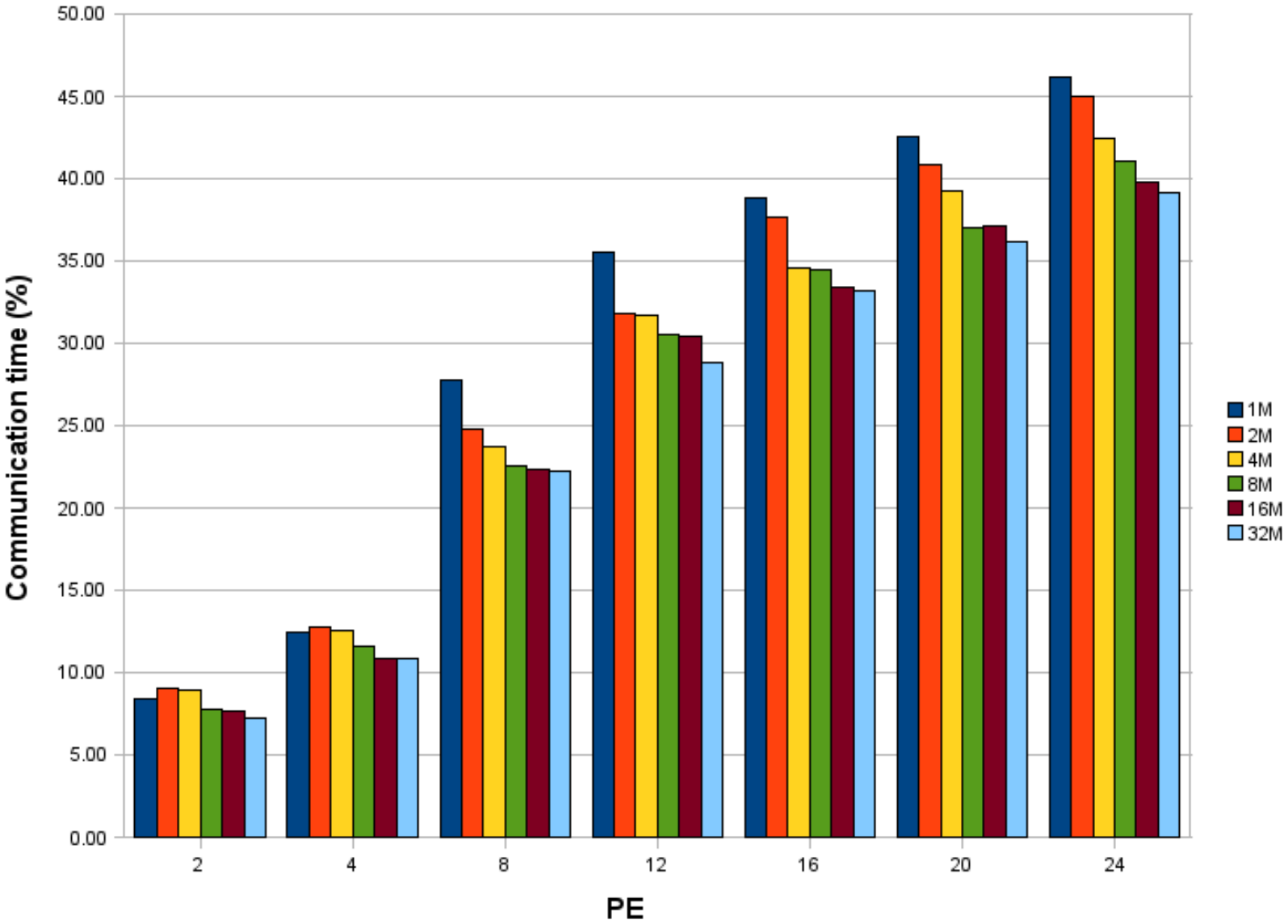
Cosa deve esserci nella relazione?

- Breve descrizione dell'algoritmo parallelo
- Prestazioni sia assolute che relative
- Grafici con la scalabilità per le diverse taglie di input
- Grafici con i rapporti fra calcolo e comunicazioni
- Confronti cambiando la configurazione (con o senza shared memory, usando la rete ethernet anziché lo switch, variando le primitive, ecc. ecc.)
- Non mettete 1000 tabelle piene di dati se non le spiegate
- Analisi delle prestazioni: i.e. perché l'algoritmo si comporta cosí, quali sono i colli di bottiglia

Cosa deve esserci nella relazione?
Grafico con la scalabilità



Cosa deve esserci nella relazione?
Peso delle comunicazioni



Altro software utile

Accesso remoto e copia in Windows

- PuTTY e PSCP: interfaccia semplice e leggera
- Secure Shell Client: interfaccia completa, drag-and-drop
- Per il download:
  `http://www.dei.unipd.it/~addetto/ssh/index.html`

Implementazioni free di MPI

OpenMPI `http://www.open-mpi.org` (no Windows)

MPICH `http://www-unix.mcs.anl.gov/mpi/mpich1` (anche Windows)

Riferimenti

- Per lo standard:
  `http://www.mpi-forum.org/docs`
- Tutorial:
  `https://computing.llnl.gov/tutorials/mpi`
- Per l'implementazione IBM:
  `http://www.dei.unipd.it/~addetto/manuali_online/index.html`
- In particolare:
  `http://www.redbooks.ibm.com/redbooks/pdfs/sg245380.pdf`