NEC HPC platforms

Introduction and motivation

H. Berger  NEC/ESS
EHPCTC Stuttgart

Why are you here?

- Because your simulation requires
  - an extraordinary amount of memory
  - an extraordinary amount of CPU time
  - an extraordinary amount of disk space or I/O performance
- Because you want to learn to write parallel code
- Because you want to learn to get the maximum out of your code?
Where does performance come from?

- 50%: fast systems (from NEC...)
- other 50%: fast code (from you...)
- key performance enablers are
  - parallelism ➔ pipelining
  - parallelism ➔ superscalar design
  - parallelism ➔ multiple CPUs
  - bandwidth
  - clock rate

How to get performance

- writing fast code is writing parallel code
- writing parallel code does not start with MPI or OpenMP
- single thread performance should be improved first
- your goal is not scalability, but time to solution!
- learn to exploit lower levels of parallelism
- make it visible - the compiler will make the rest
Understand and benefit

- By knowing where performance comes from, you can learn where performance disappears
- try to understand your hardwares architecture
- what can you expect?
- What do you get?
- Why is it not the same?
- Next step: improve your algorithms

Clock rate

- Simple: the higher, the better
- but: can memory keep up?
- What to do with several billion operations per second on only 100 million operands?
- Solution: caches
- fast, expensive and small memory
- you are lucky if your data fits in
- otherwise, you are lost. Life is that simple.
Bandwidth

- As mentioned: cannot keep up with clockrate growth
- Bandwidth and latency are closely related
- Latency is even worse, as it is not decreasing
- Bandwidth is determined by
  - Bus clock speed
  - Bus width
  - Latency
  - Number of outstanding transactions

Pipelining

- Very simple and well known approach to speed up tasks consisting of subtasks
- Example: automotive industry
  - Move the car
  - Every pipeline stage makes the car more complete
  - Every stage is specialized for one task
Pipelining 2

- Works only if operations are independant
- it takes the same time to get the result
- but: more results can be computed in the same time
- used for long time in every days work
- used in computers for >30 years
- is used in PCs for ~ 10 years
- NEC SX vector computer: pipelined everything
  - computation
  - memory access ➔ latency hiding!

Superscalar design

- Simply add arithmetic units
- for example: two multiply-add unions instead of one
- to keep it running: several independant operations have to be available
- available for ~ 10 years in PCs
- in SX series: 8 or 16 parallel sets of pipelines
- in Azusa: 2 multiply-add units
Several cpus

- Two ways: „shared memory“ or „distributed memory“
- shared memory offers high comfort
- incremental parallelization is possible
- drawback: higher costs
- solutions:
  - distributed shared memory
  - non uniform access shared memory

Small vs. Big

- Last 10 years: trend away from single strong CPU towards many weak CPU
- Promise: cheaper and as fast as vector
- Problem: Amdahls law
- Just adding hardware does not solve the problem
- Software has to improve as well
- Can software improve enough?
- Can YOU improve your software enough?
Why strong single CPU?

- Amdahls law

<table>
<thead>
<tr>
<th></th>
<th>98</th>
<th>99</th>
<th>99.90</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>7.02</td>
<td>7.48</td>
<td>7.94</td>
</tr>
<tr>
<td>16</td>
<td>12.31</td>
<td>13.91</td>
<td>15.76</td>
</tr>
<tr>
<td>512</td>
<td>45.63</td>
<td>83.80</td>
<td>338.85</td>
</tr>
<tr>
<td>1024</td>
<td>47.72</td>
<td>91.18</td>
<td>506.18</td>
</tr>
</tbody>
</table>

- Might be a good idea to operate in the „nice“ area of amdahls law...

Distributed shared memory

- Example: NEC SX series
- nodes with up to 16 CPUs with up to 128 GB of shared memory
- can be coupled to a cluster using IXS crossbar
- programming model:
  - Thread parallelism inside node
  - message passing between nodes over IXS link
Non uniform shared memory

- Example: NEC AzusA
- up to 16 CPUs on 64 GB shared memory
- system consists of 4 cells with 4 CPUs each
- cells are connected by crossbar
- cache coherency is done by hardware
- remote latency is very low
- feels like uniform shared memory

Scalar vs vector

- Modern RISC CPU
  - pipelining
  - superscalar
  - software pipelining
  - high bandwidth caches
- Modern vector CPU
  - vector pipelines
  - several pipe sets
  - chaining
  - vector data registers
  - high bandwidth memory
  - pipelined memory access

- RISC learned a lot from vector computers
- But they suffer from bandwidth due to non pipelined memory access
Scalar vs vector

- E-pipelines effective pipelines, due to higher clock rate

Why NEC?

- All key technologies for HPC inside NEC
  - Semiconductor Devices
  - Packaging
  - HW Design
  - Interconnections and Network
  - Operating Systems Software
  - Languages and Tools
  - Applications Tuning and Support
NEC products for HPC

Vector Supercomputers

IA-64/Linux Platform

• CFD
• Crash Analysis
• Structural Analysis
• Mechanical Design
• Technical Computing
• Engineering

Windows Workstations

• CAD
• EDA
• Graphics

Products Enhancements Released

Peak Performance

<table>
<thead>
<tr>
<th>(FLOPS)</th>
<th>CPU Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>8G - 16GFLOPS</td>
<td>HPC Server SX-5S</td>
</tr>
<tr>
<td>4G - 8GFLOPS</td>
<td>SX-5 Series</td>
</tr>
</tbody>
</table>

SX-5 Series

IXS or HIPPI-SW

A Model
64G - 128GF

B Model
32G - 64GFLOPS

C Model
16G - 32GFLOPS

D Model
8G - 16GFLOPS

NEC

24.

NEC HPC Platforms

24-10
Processor Card

457 x 386mm
225 x 225 mm
20 x 20mm

- SX-4
  - Performance: 2 GFlops, 8 nS
  - LSI: 0.35 μm CMOS
  - 37 Chips

- SX-5
  - Performance: 8 GFlops, 4 nS
  - 0.25 μm CMOS
  - 32 Chips

- SX-5X
  - Performance: 16 GFlops, 2 nS
  - 0.18 μm CMOS
  - 1 Chip Processor

Memory Card

457 x 386mm
457 x 386
105 x 120mm

- SX-4
  - Capacity: 256MB / Card
  - Memory Chip: 256MB SRAM

- SX-5
  - 4GB / Card
  - 64MB SRAM

- SX-5X
  - 1GB / Card
  - 256MB SRAM
Questions?

- Support desk at HWW:

  hwwsupport@ess.nec.de