High-Performance Computing Today

Jack Dongarra
Innovative Computing Laboratory
University of Tennessee
and
Oak Ridge National Laboratory
http://www.cs.utk.edu/~dongarra/

Outline

? Look at trends in HPC
  ✷ Top500 statistics
? Performance of Super-Scalar Processors
  ✷ ATLAS
? Performance Monitoring
  ✷ PAPI
? NetSolve
  ✷ Example of grid middleware

In pioneer days, they used oxen for heavy pulling, and when one ox couldn’t budge a log they didn’t try to grow a larger ox. We shouldn’t be trying for bigger computers, but for more systems of computers. -- Grace Hopper
Technology Trends: Microprocessor Capacity

2X transistors/Chip Every 1.5 years
Called “Moore’s Law”

Microprocessors have become smaller, denser, and more powerful.

Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

High Performance Computers & Numerical Libraries

\[ 20 \text{ years ago} \]
- \(1 \times 10^6\) Floating Point Ops/sec (Mflop/s)
  - Scalar based
  - Loop unrolling

\[ 10 \text{ years ago} \]
- \(1 \times 10^9\) Floating Point Ops/sec (Gflop/s)
  - Vector & Shared memory computing, bandwidth aware
  - Block partitioned, latency tolerant

\[ \text{Tod}}\]
- \(1 \times 10^{12}\) Floating Point Ops/sec (Tflop/s)
  - Highly parallel, distributed processing, message passing, network based
  - data decomposition, communication/computation

\[ 10 \text{ years away} \]
- \(1 \times 10^{15}\) Floating Point Ops/sec (Pflop/s)
  - Many more levels MH, combination/grids\&HPC
  - More adaptive, LT and bandwidth aware, fault tolerant, extended precision, attention to SMP nodes
**TOP500**

- Listing of the 500 most powerful Computers in the World
- Yardstick: Rmax from LINPACK MPP
  \[ Ax = b, \text{ dense problem} \]
- Updated twice a year
  SC‘xy in the States in November
  Meeting in Mannheim, Germany in June
- All data available from [www.top500.org](http://www.top500.org)

---

**Fastest Computer Over Time**

In 1980 a computation that took 1 full year to complete can now be done in 1 month!
In 1980 a computation that took 1 full year to complete can now be done in 4 days!

Year

GFlop/s

Fastest Computer Over Time

Hitachi
CP-PACS
(int2040)

Intel
Paragon
(int6788)

Fujitsu
VPP-500

TMC
CM-5
(int140)

NEC
SX-3
(int1024)

In 1980 a computation that took 1 full year to complete can now be done in 1 hour!
### Top 10 Machines (June 2000)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Company</th>
<th>Machine</th>
<th>Procs</th>
<th>Gflop/s</th>
<th>Place</th>
<th>Country</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Intel</td>
<td>ASCI Red</td>
<td>9632</td>
<td>2380</td>
<td>Sandia National Labs</td>
<td>USA</td>
<td>1999</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Albuquerque</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>IBM</td>
<td>ASCI Blue-Pacific SST, IBM SP 604e</td>
<td>5808</td>
<td>2144</td>
<td>Lawrence Livermore National Laboratory Livermore</td>
<td>USA</td>
<td>1999</td>
</tr>
<tr>
<td>3</td>
<td>SGI</td>
<td>ASCI Blue Mountain</td>
<td>6144</td>
<td>1608</td>
<td>Los Alamos National Laboratory Los Alamos</td>
<td>USA</td>
<td>1998</td>
</tr>
<tr>
<td>4</td>
<td>Hitachi</td>
<td>SR8000-F1/112</td>
<td>112</td>
<td>1035</td>
<td>Leibniz Rechenzentrum Muenchen</td>
<td>Germany</td>
<td>2000</td>
</tr>
<tr>
<td>5</td>
<td>Hitachi</td>
<td>SR8000-F1/100</td>
<td>100</td>
<td>917</td>
<td>High Energy Accelerator Research Organization /KEK Tsukuba</td>
<td>Japan</td>
<td>2000</td>
</tr>
<tr>
<td>6</td>
<td>Cray Inc.</td>
<td>T3E1200</td>
<td>1084</td>
<td>892</td>
<td>Government</td>
<td>USA</td>
<td>1998</td>
</tr>
<tr>
<td>7</td>
<td>Cray Inc.</td>
<td>T3E1200</td>
<td>1084</td>
<td>892</td>
<td>US Army HPC Research Center at NCS Minneapolis</td>
<td>USA</td>
<td>2000</td>
</tr>
<tr>
<td>8</td>
<td>Hitachi</td>
<td>SR8000/128</td>
<td>128</td>
<td>874</td>
<td>University of Tokyo Tokyo</td>
<td>Japan</td>
<td>1999</td>
</tr>
<tr>
<td>9</td>
<td>Cray Inc.</td>
<td>T3E900</td>
<td>1324</td>
<td>815</td>
<td>Government</td>
<td>USA</td>
<td>1997</td>
</tr>
<tr>
<td>10</td>
<td>IBM</td>
<td>SP Power3 375 MHz</td>
<td>1336</td>
<td>723</td>
<td>Naval Oceanographic Office (NAVOCEANO) Poughkeepsie</td>
<td>USA</td>
<td>2000</td>
</tr>
</tbody>
</table>

### Performance Development

- **64.3 TF/s**
- **2.38 TF/s**
- **39.9 GF/s**
- **100 Mflop/s**
- **1 Gflop/s**
- **10 Gflop/s**
- **100 Gflop/s**
- **1 TFlop/s**
- **10 TFlop/s**
- **100 TFlop/s**

- **SUM**
- **N=1**
- **N=500**

- Intel ASCI Red
- Sun Ultra HPC 1000
- IBM 6046 69 proz.

- Fujitsu "NWT" NAL
- SGI POWER CHALLENGE GOODYEAR
- Sun HPC 10000 Merrill Lynch
- NEC SX-5 Supercomputer
- Cray T3D MP CM4/324 "EPA" USA

- Intel XP/S140 Sandia
- Hitachi/Tsukuba CP PACS/3048
- Cray Y-MP M944 USA
- Sparc Ultra 2000

- "Fast than Moore's Law"
High-Performance Computing Directions:
Beowulf-class PC Clusters

**Definition:**
- COTS PC Nodes
  - Pentium, Alpha, PowerPC, SMP
- COTS LAN/SAN Interconnect
  - Ethernet, Myrinet, Giganet, ATM
- Open Source Unix
  - Linux, BSD
- Message Passing Computing
  - MPI, PVM
  - HPF

**Advantages:**
- Best price-performance
- Low entry-level cost
- Just-in-place configuration
- Vendor invulnerable
- Scalable
- Rapid technology tracking

Enabled by PC hardware, networks and operating system achieving capabilities of scientific workstations at a fraction of the cost and availability of industry standard message passing libraries.
Where Does the Performance Go? or Why Should I Care About the Memory Hierarchy?

Processor-DRAM Memory Gap (latency)

**Processor-Memory Performance Gap:** (grows 50% / year)

**Moore’s Law**

**μProc** 60%/yr. (2X/1.5yr)

**DRAM** 9%/yr. (2X/10 yrs)

---

Optimizing Computation and Memory Use

**Computational optimizations**

- **Theoretical peak:**
  - \((\# \text{ fpus}) \times (\text{flops/cycle}) \times \text{Mhz}\)
  - PIII: \((1 \text{ fpu}) \times (1 \text{ flop/cycle}) \times (650 \text{ Mhz}) = 650 \text{ MFLOP/s}\)
  - Athlon: \((2 \text{ fpu}) \times (1 \text{ flop/cycle}) \times (600 \text{ Mhz}) = 1200 \text{ MFLOP/s}\)
  - Power3: \((2 \text{ fpu}) \times (2 \text{ flops/cycle}) \times (375 \text{ Mhz}) = 1500 \text{ MFLOP/s}\)

**Memory optimization**

- **Theoretical peak:** (bus width) \(\times\) (bus speed)
  - PIII: \((32 \text{ bits}) \times (133 \text{ Mhz}) = 532 \text{ MB/s} = 66.5 \text{ MW/s}\)
  - Athlon: \((64 \text{ bits}) \times (200 \text{ Mhz}) = 1600 \text{ MB/s} = 200 \text{ MW/s}\)
  - Power3: \((128 \text{ bits}) \times (100 \text{ Mhz}) = 1600 \text{ MB/s} = 200 \text{ MW/s}\)

**Memory about an order of magnitude slower**
Memory Hierarchy

- By taking advantage of the principle of locality:
  - Present the user with as much memory as is available in the cheapest technology.
  - Provide access at the speed offered by the fastest technology.

<table>
<thead>
<tr>
<th>Level</th>
<th>Speed (ns)</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>1s</td>
<td>100s</td>
</tr>
<tr>
<td>Datapath</td>
<td>10s</td>
<td>Ks</td>
</tr>
<tr>
<td>Main Memory (DRAM)</td>
<td>100s</td>
<td>Ms</td>
</tr>
<tr>
<td>Level 2 and 3 Cache (SRAM)</td>
<td>10,000,000s (10s ms)</td>
<td>Gs</td>
</tr>
<tr>
<td>Secondary Storage (Disk)</td>
<td>10,000,000s (10s sec)</td>
<td>Ts</td>
</tr>
<tr>
<td>Tertiary Storage (Disk/Tape)</td>
<td>10,000,000 s (10s ms)</td>
<td>Ts</td>
</tr>
</tbody>
</table>

How To Get Performance From Commodity Processors?

- Today’s processors can achieve high-performance, but this requires extensive machine-specific hand tuning.
- Hardware and software have a large design space w/many parameters
  - Blocking sizes, loop nesting permutations, loop unrolling depths, software pipelining strategies, register allocations, and instruction schedules.
  - Complicated interactions with the increasingly sophisticated micro-architectures of new microprocessors.
- Until recently, no tuned BLAS for Pentium for Linux.
- Need for quick/dynamic deployment of optimized routines.
- ATLAS - Automatic Tuned Linear Algebra Software
  - PhiPac from Berkeley
  - FFTW from MIT (http://www.fftw.org)
ATLAS

- An adaptive software architecture
  - High-performance
  - Portability
  - Elegance

- ATLAS is faster than all other portable BLAS implementations and it is comparable with machine-specific libraries provided by the vendor.

ATLAS Across Various Architectures (DGEMM n=500)

- ATLAS is faster than all other portable BLAS implementations and it is comparable with machine-specific libraries provided by the vendor.
Code Generation Strategy

- On-chip multiply optimizes for:
  - TLB access
  - L1 cache reuse
  - FP unit usage
  - Memory fetch
  - Register reuse
  - Loop overhead minimization
- Takes a 30 minutes to a hour to run.
- New model of high performance programming where critical code is machine generated using parameter optimization.

- Code is iteratively generated & timed until optimal case is found.
- We try:
  - Differing NBs
  - Breaking false dependencies
  - M, N and K loop unrolling

- Designed for RISC arch
  - Super Scalar
  - Need reasonable C compiler

Plans for ATLAS

- Software Release, available today:
  - Level 1, 2, and 3 BLAS implementations
  - See: www.netlib.org/atlas/

- Next Version:
  - Multi-threading
  - Java generator

- Futures:
  - Optimize message passing system
  - Runtime adaptation
    » Sparsity analysis
    » Iterative code improvement
  - Specialization for user applications
  - Adaptive libraries
Tools for Performance Evaluation

- Timing and performance evaluation has been an art
  - Resolution of the clock
  - Issues about cache effects
  - Different systems
- Situation about to change
  - Today's processors have internal counters

Performance Counters

- Almost all high performance processors include hardware performance counters.
- Some are easy to access, others not available to users.
- On most platforms the APIs, if they exist, are not appropriate for a common user, functional or well documented.
- Existing performance counter APIs
  - Cray T3E
  - SGI MIPS R10000
  - IBM Power series
  - DEC Alpha pfm pseudo-device interface
  - Windows 95, NT and Linux
Performance Data That May Be Available

- Cycle count
- Floating point instruction count
- Integer instruction count
- Instruction count
- Load/store count
- Branch taken / not taken count
- Branch mispredictions
- Pipeline stalls due to memory subsystem
- Pipeline stalls due to resource conflicts
- I/D cache misses for different levels
- Cache invalidations
- TLB misses
- TLB invalidations

PAPI’s Graphical Tools
Perfometer Usage

? Application is instrumented with PAPI
  - call perfometer()

? Will be layered over the best existing vendor-specific APIs for these platforms

? Sections of code that are of interest are designated with specific colors
  - Using a call to set_perfometer('color')

? Application is started, at the call to perfometer a task is spawned to collect and send the information to a Java applet containing the graphical view.
Perfometer

Machine info

Flops issued

Flop/s Rate

Flop/s Instantaneous Rate

Process & Real time

Call Perfometer('red')

Go To Demo
Trends in Computational Science and Engineering

- Multi-scale, Multi-physics, Multi-dimensional simulations of realistic complexity
- Growing use of dynamic adaptive algorithms
- Strong interplay between observation and simulation (e.g., cosmology, weather)
- Impact of the WWW
  - accelerated pace of research due to electronic publishing
  - proliferation of digital archives
  - emergence of workbenches and portals

Grid Computing

- To treat CPU cycles and software like commodities, an application should be:
  - Ubiquitous -- able to interface to the system at any point and leverage whatever is available
  - Resource Aware -- capable of managing heterogenity
  - Adaptive -- able to tailor its behavior dynamically so that it gets maximum performance benefit from the services and resources at hand
The Grid Architecture Picture

User Portals
- Grid Access & Info
- Problem Solving Environments
- Application Science Portals

Service Layers
- Co-Scheduling
- Resource Discovery & Allocation
- Fault Tolerance
- Authentication
- Naming & Files
- Events

Resource Layer
- computers
- Data bases
- Online instruments
- High speed networks and routers

System Users
Scientists and engineers using computation to accomplish lab missions.

Intelligent Interface
A knowledge-based environment that offers users guidance on complex computing tasks.

Middleware
Software tools that enable interaction among users, applications, and system resources.

Cluster Operating System
The software which coordinates the interplay of computers, networks and storage.

Supercomputing
Heterogeneous collection of high-performance computer hardware and software resources.

Networking
The hardware and software that permits communication among distributed users and computer resources.

Mass Storage
A collection of devices and software that allow temporary and long-term archival storage of information.
Motivation for NetSolve

Design an easy-to-use tool to provide efficient and uniform access to a variety of scientific packages on UNIX and Windows platforms

Basics

- Client-Server Design
- Non-hierarchical system
- Load Balancing and Fault Tolerance
- Heterogeneous Environment Supported
- Multiple and simple client interfaces
- Built on standard components

NetSolve - The Big Picture

Computational Resources

Hardware: Software:
Clusters: Routines
MPP: Libraries
Workstations: Applications
Globus, Condor, MPPVM

Request

Agent

Scheduler
Database

No knowledge of the grid required

Matlab
Mathematica
C, Fortran
Java, Excel
Java GUI

Client - RPC like

Reply

Choice
NetSolve

Three deployment scenarios:
- Client, servers and agents anywhere on Internet
- Client, servers and agents on an Intranet
- Client, server and agent on the same machine

“Blue Collar” Grid Based Computing
- User can set things up, no “su” required
- Does not require deep knowledge of network programming

Smart Libraries
- “Rent” access to routines
- Decouple interface

NetSolve Usage Scenarios

- Grid based library routines
  - Users doesn’t have to have library routines on their machine

- Task farming applications
  - “Pleasantly parallel” execution
  - eg Parameter studies

- Remote application execution
  - Complete packages with user specifying input parameters
NetSolve - MATLAB Interface

**Synchronous Call**

```
>> define sparse matrix A
>> define rhs
>> [x, its] = netsolve('itmeth','petsc', A, rhs, 1.e-6, 50);
```

```
>> [x, its] = petsc(A, rhs);   % for PETSc
>> [x, its] = aztec(A, rhs);   % for AZTEC
>> [x] = superlu(A, rhs);     % for SuperLU
>> [x] = ma28(A, rhs);        % for MA28
```

Asynchronous Calls also available

NetSolve - FORTRAN Interface

**Easy to ‘switch’ to NetSolve**

```fortran
parameter( MAX = 100)
double precision A(MAX,MAX), B(MAX)
integer IPIV(MAX), N, INFO, LWORK
integer NSINFO

call DGESV(N,1,A,MAX,IPIV,B,MAX,INFO)

call NETSL(‘DGESV()’,NSINFO,
           N,1,A,MAX,IPIV,B,MAX,INFO)
```
Hiding the Parallel Processing

User maybe unaware of parallel processing

NetSolve takes care of the starting the message passing system, data distribution, and returning the results.

MCell: 3-D Monte-Carlo Simulation of Neurotransmitter Release in Between Cells

- Developed at: Salk Institute (T. Bartol), Cornell U. (J. Stiles)
- Study how neurotransmitters diffuse and activate receptors in synapses
- blue unbounded, red singly bounded, green doubly bounded closed, yellow doubly bounded open
Integrated Parallel Accurate Reservoir Simulator.

Mary Wheeler's group, UT-Austin

Reservoir and Environmental Simulation.

- models black oil, waterflood, compositions
- 3D transient flow of multiple phase

Integrates Existing Simulators.

Framework simplified development

- Provides solvers, handling for wells, table lookup.
- Provides pre/postprocessor, visualization.

Full IPARS access without Installation.

IPARS Interfaces Now Available:

- C, FORTRAN, Matlab, Mathematica, and Web.

NetSolve Applications and Interactions

Tool integration

- Globus - Middleware infrastructure (ANL/SSI)
- Condor - Workstation farm (U Wisconsin)
- NWS - Network Weather Service (U Tennessee)
- SCIRun - Computational steering (U Utah)
- Ninf - NetSolve-like system, (ETL,Tsukuba)

Library usage

- LAPACK/ScalAPACK - Parallel dense linear solvers
- SuperLU/MA28 - Parallel sparse direct linear solvers(UCB/RAL)
- PETSc/Aztec - Parallel iterative solvers (ANL/SNL)
- Other areas as well (not just linear algebra)

Applications

- MCell - Microcellular physiology (UCSD/Salk)
- IPARS - Reservoir Simulator (UTexas, Austin)
- Virtual Human - Pulmonary System Model (ORNL)
- RSICC - Radiation Safety sw/simulation (ORNL)
- LUCAS - Land usage modeling (U Tennessee)
- ImageVision - Computer Graphics and Vision (Graz U)
Conclusion

- Exciting time to be in scientific computing
- Network computing is here
- The Grid offers tremendous opportunities for collaboration
- Important to develop algorithms and software that will work effectively in this environment

Contributors to These Ideas

- Top500
  - Erich Strohmaier, UTK
  - Hans Meuer, Mannheim U
- ATLAS
  - Antoine Petit, UTK
  - Clint Whaley, UTK
- PAPI
  - Shirley Browne, UTK
  - Nathan Garner, UTK
  - Kevin London, UTK
  - Phil Mucci, UTK
- NetSolve
  - Dorian Arnold, UTK
  - Susan Blackford, UTK
  - Henri Casanova, UCSD
  - Michelle Miller, UTK
  - Sathish Vadhiyar, UTK

For additional information see...

- www.netlib.org/top500/
- www.netlib.org/atlas/
- icl.cs.utk.edu/projects/papi/
- www.netlib.org/netsolve/
- www.cs.utk.edu/~dongarra/

Many opportunities within group