Overview

- Introduction to
  - JUMP
  - the SPEC MPI 2007 suite
- SCALASCA overview
- Measurement methodology
- Experimental results
Motivation

- SPEC MPI 2007 provides a representative set of MPI application kernels
  - 13 applications
  - C/C++/Fortran, often combined
  - Different MPI programming patterns
    - blocking/nonblocking/persistent communication
    - with or without extensive collective communication
  - Most of them show iterative behavior suitable for phase instrumentation
JUMP - IBM SP2 Regatta p690+ system

- 41 IBM SP2 p690+ frames
  - 16 dual-core 1.7GHz Power4+ processors
  - 128GB of shared main memory
- Connected via IBM High Performance Switch
- GPFS filesystem
- AIX 5.3
- IBM XL compiler suites
  - version 7.0/8.0 for C/C++
  - version 9.1/10.1 for Fortran
- IBM POE 4.2 MPI
## SPEC MPI 2007 applications overview

<table>
<thead>
<tr>
<th>Code</th>
<th>LOC</th>
<th>Language</th>
<th>Application subject area</th>
</tr>
</thead>
<tbody>
<tr>
<td>104.milc</td>
<td>17987</td>
<td>C</td>
<td>Lattice quantum chromodynamics</td>
</tr>
<tr>
<td>107.leslie3d</td>
<td>10503</td>
<td>F77, F90</td>
<td>Combustion dynamics</td>
</tr>
<tr>
<td>113.GemsFDTD</td>
<td>21858</td>
<td>F90</td>
<td>Computational electrodynamics</td>
</tr>
<tr>
<td>115.fds4</td>
<td>44524</td>
<td>F90, C</td>
<td>Computational fluid dynamics</td>
</tr>
<tr>
<td>121.pop2</td>
<td>69203</td>
<td>F90</td>
<td>Oceanography</td>
</tr>
<tr>
<td>122.tachyon</td>
<td>15512</td>
<td>C</td>
<td>Computer graphics: ray tracing</td>
</tr>
<tr>
<td>126.lammps</td>
<td>6796</td>
<td>C++</td>
<td>Molecular dynamics</td>
</tr>
<tr>
<td>127.wrf2</td>
<td>163462</td>
<td>F90, C</td>
<td>Numerical weather prediction</td>
</tr>
<tr>
<td>128.GAPgeofem</td>
<td>30935</td>
<td>F77, C</td>
<td>Geophysics finite-element methods</td>
</tr>
<tr>
<td>129.tera_tf</td>
<td>6468</td>
<td>F90</td>
<td>Eulerian hydrodynamics</td>
</tr>
<tr>
<td>130.socorro</td>
<td>91585</td>
<td>F90</td>
<td>Quantum chemistry</td>
</tr>
<tr>
<td>132.zeusmp2</td>
<td>44441</td>
<td>C, F90</td>
<td>Astrophysical hydrodynamics</td>
</tr>
<tr>
<td>137.lu</td>
<td>5671</td>
<td>F90</td>
<td>Linear algebra SSOR</td>
</tr>
</tbody>
</table>
SPEC MPI 2007 execution times on JUMP
The SCALASCA project

Overview

- Started in January 2006
- Funded by Helmholtz Initiative & Networking Fund
- Developed in collaboration with ICL/UT
- Follow-up to pioneering KOJAK project
  - Automatic pattern-based performance analysis

Objective

- Development of a scalable performance analysis toolset
- Specifically targeting large-scale applications
Features

Open source

Portable

- BG/L, BG/P, IBM SP & blade clusters, Cray XT3/4, SGI Altix, Sun Fire clusters (SPARC, x86-64), ...

Supports various languages & parallel programming paradigms

- Fortran, C, C++
- MPI, OpenMP & hybrid MPI/OpenMP

Integrated measurement & analysis toolset

- Runtime summarization (aka profiling)
- Automatic event trace analysis
Application instrumentation

- Automatic/manual code instrumenter
  - Processes program sources
  - Adds instrumentation and measurement library into application executable
- Measurement library
  - Exploits MPI standard profiling interface (PMPI)
  - Provides measurement infrastructure & instrumentation API
Runtime summarization

- Measurement library manages threads & events produced by instrumentation
- Measurements summarized by thread & call-path during execution
- Analysis report unified & collated at finalization
- Presentation of analysis results
Event tracing & analysis

- Time-stamped events buffered during measurement for each thread
- Flushed to files along with unified definitions & mapping tables at finalization
- Follow-up analysis replays events and produces extended analysis report
- Presentation of analysis report
SCALASCA components

- Automatic/manual code instrumenter
- Unified measurement library supporting both
  - runtime summaries
  - trace file generation
- Parallel, replay-based trace analyzer
- Common analysis report examiner

Diagram:

1. Program sources
   - Compiler
   - Instrumenter
   - Instrumented executable

2. Application + measurement lib
   - Unified defs+maps
   - Trace 1 2 .. N

3. Parallel trace analyzer
   - Summary analysis
   - Trace analysis

4. Analysis report examiner
   - Expt config
Runtime summary report

- What kind of performance problem?
- Where is it in the source code? In what context?
- How is it distributed across the system?
132.zeusmp2 point-to-point communication time

- Summary experiment
- 12.8% of time spent in MPI point-to-point communication
- 13.9% of which is on a single callpath to an MPI_Waitall
- Has a 35.6% standard variation across 512 processes
- Highest values in 2\textsuperscript{nd} and 7\textsuperscript{th} planes of the Cartesian grid
Trace analysis report

Additional trace-based metrics

No hardware counters
Phase-instrumentation

- Motivation
  - *This is all very good, but I still can't see how things evolve over time*
  - *Is there a way to analyze time-dependent behavior?*

- Phase-instrumentation
  - *Identify the main loop of the application*
  - *Place markers at the beginning and the end of the loop*
  - *SCALASCA will now save all the iterations separately*
132.zeusmp2 late sender time

- Tracing experiment
- Including manually inserted timestep annotations
- Neighbors are waiting for a few slower processes in the first timestep
Measurement & analysis methodology

1. Run uninstrumented version (as reference for validation)
   - Determine available memory from heap usage

2. Run fully-instrumented version collecting runtime summary
   - Determine functions with excessive overheads
     - *(distortion and trace buffer cost)*
   - Prepare filter file

3. Repeat measurement with filter
   - Verify dilation and trace buffer cost
   - Identify appropriate iterations/timesteps to be annotated

4. Repeat summarization with filter and phases

5. Reconfigure measurement to collect and analyze traces (with phases)
   - Generate graphs and charts of metrics per iteration/timestep
### SPEC MPI 2007 32-way execution characteristics

<table>
<thead>
<tr>
<th>Code</th>
<th>Program execution</th>
<th>RSS (MB)</th>
<th>Trace buffer content (MB/proc)</th>
<th>Filter funcs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>steps  depth  callpaths</td>
<td>total</td>
<td>MPI    filter   residue</td>
<td></td>
</tr>
<tr>
<td>104.milc</td>
<td>8+243 6 257</td>
<td>341</td>
<td>2683   1.7   2626   57</td>
<td>4</td>
</tr>
<tr>
<td>107.leslie3d</td>
<td>2000 3 40</td>
<td>1078</td>
<td>1437   15.0  1422  16</td>
<td>6</td>
</tr>
<tr>
<td>113.GemsFDTD</td>
<td>1000 5 185</td>
<td>505</td>
<td>3619   5.9   3582   37</td>
<td>1</td>
</tr>
<tr>
<td>115.fds4</td>
<td>2363 8 151</td>
<td>209</td>
<td>122    2.1   117    6</td>
<td>6</td>
</tr>
<tr>
<td>121.pop2</td>
<td>9000 6 403</td>
<td>748</td>
<td>6361   2494.0 2606  3841 6</td>
<td></td>
</tr>
<tr>
<td>122.tachyon</td>
<td>N/A 3 27</td>
<td>676</td>
<td>59884  0.7   59809  75</td>
<td>5</td>
</tr>
<tr>
<td>126.lammps</td>
<td>500 6 162</td>
<td>401</td>
<td>291    0.8   290    1</td>
<td>9</td>
</tr>
<tr>
<td>127.wrf2</td>
<td>1375 22 4975</td>
<td>297</td>
<td>1109   0.4   1106  5</td>
<td>69</td>
</tr>
<tr>
<td>128.GAPgeofem</td>
<td>235 4 44</td>
<td>361</td>
<td>996    33.0  971    34</td>
<td>2</td>
</tr>
<tr>
<td>129.tera_tf</td>
<td>943 4 59</td>
<td>74</td>
<td>2459   10.0  1628  831</td>
<td>4</td>
</tr>
<tr>
<td>130.socorro</td>
<td>20 23 10352</td>
<td>148</td>
<td>10703  13.0  10587 120</td>
<td>21</td>
</tr>
<tr>
<td>132.zeusmp2</td>
<td>200 5 179</td>
<td>377</td>
<td>5      3.4   -      3</td>
<td>0</td>
</tr>
<tr>
<td>137.lu</td>
<td>180 4 49</td>
<td>384</td>
<td>42     28.0  -      28</td>
<td>0</td>
</tr>
</tbody>
</table>
SPEC MPI 2007 32-way execution times for different instrumentation configurations in wallclock seconds

<table>
<thead>
<tr>
<th>Code</th>
<th>Instrumentation/measurement</th>
<th>Trace flushing</th>
<th>Trace analysis</th>
<th>Trace size (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Sum</td>
<td>Sum+pf</td>
<td>Trace</td>
<td></td>
</tr>
<tr>
<td>104.milc</td>
<td>1556</td>
<td>2140</td>
<td>1616</td>
<td>1611</td>
</tr>
<tr>
<td>107.leslie3d</td>
<td>2704</td>
<td>2945</td>
<td>2807</td>
<td>2892</td>
</tr>
<tr>
<td>113.GemsFDTD</td>
<td>2028</td>
<td>2680</td>
<td>2042</td>
<td>2111</td>
</tr>
<tr>
<td>115.fds4</td>
<td>951</td>
<td>1010</td>
<td>960</td>
<td>957</td>
</tr>
<tr>
<td>121.pop2</td>
<td>1687</td>
<td>2415</td>
<td>2176</td>
<td>2104</td>
</tr>
<tr>
<td>122.tachyon</td>
<td>2024</td>
<td>N/P</td>
<td>6016</td>
<td>6023</td>
</tr>
<tr>
<td>126.lammps</td>
<td>1883</td>
<td>1988</td>
<td>1899</td>
<td>2001</td>
</tr>
<tr>
<td>127.wrf2</td>
<td>2352</td>
<td>2945</td>
<td>2499</td>
<td>2475</td>
</tr>
<tr>
<td>128.GAPgeofem</td>
<td>833</td>
<td>984</td>
<td>879</td>
<td>884</td>
</tr>
<tr>
<td>129.tera_tf</td>
<td>2399</td>
<td>2583</td>
<td>2458</td>
<td>2395</td>
</tr>
<tr>
<td>130.socorro</td>
<td>1411</td>
<td>3990</td>
<td>1631</td>
<td>1701</td>
</tr>
<tr>
<td>132.zeusmp2</td>
<td>1683</td>
<td>1727</td>
<td>-</td>
<td>1729</td>
</tr>
<tr>
<td>137.lu</td>
<td>1771</td>
<td>1815</td>
<td>-</td>
<td>1910</td>
</tr>
</tbody>
</table>
# SPEC MPI 2007 32-way execution time overheads for different instrumentation configurations

<table>
<thead>
<tr>
<th>Code</th>
<th>Instrumentation/measurement</th>
<th>Tracing flushing</th>
<th>Trace analysis</th>
<th>Trace size (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>104.milc</td>
<td>None</td>
<td>1556</td>
<td>13</td>
<td>1.587</td>
</tr>
<tr>
<td>107.leslie3d</td>
<td>Sum</td>
<td>2704</td>
<td>43</td>
<td>0.403</td>
</tr>
<tr>
<td>113.GemsFDTD</td>
<td>Sum+f</td>
<td>2028</td>
<td>57</td>
<td>0.634</td>
</tr>
<tr>
<td>115.fds4</td>
<td>Sum+pf</td>
<td>951</td>
<td>92</td>
<td>0.130</td>
</tr>
<tr>
<td>121.pop2</td>
<td>Trace flushing</td>
<td>1687</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>122.tachyon</td>
<td>N/P</td>
<td>2024</td>
<td>68</td>
<td>0.007</td>
</tr>
<tr>
<td>126.lammps</td>
<td>197.2%</td>
<td>1883</td>
<td>41</td>
<td>0.038</td>
</tr>
<tr>
<td>127.wrf2</td>
<td>25.2%</td>
<td>2352</td>
<td>425</td>
<td>18.138</td>
</tr>
<tr>
<td>128.GAPgeofem</td>
<td>18.1%</td>
<td>833</td>
<td>14</td>
<td>0.670</td>
</tr>
<tr>
<td>129.tera_tf</td>
<td>7.7%</td>
<td>2399</td>
<td>17</td>
<td>24.737</td>
</tr>
<tr>
<td>130.socorro</td>
<td>182.8%</td>
<td>1411</td>
<td>120</td>
<td>3.420</td>
</tr>
<tr>
<td>132.zeusmp2</td>
<td>2.6%</td>
<td>1683</td>
<td>28</td>
<td>0.113</td>
</tr>
<tr>
<td>137.lu</td>
<td>2.5%</td>
<td>1771</td>
<td>13</td>
<td>1.100</td>
</tr>
</tbody>
</table>
Iteration/timestep analysis graphs and charts

- Minimum/median/maximum value graph over iterations
- Chart of values per process color coded
  - from yellow-white (min)
  - to dark-red (max)
132.zeusmp2 execution time

- First timestep takes longest
- Progressive increase from 7.9s to 8.8s
- Transitions at timesteps 41 and 69
- A low amount of noise is visible
132.zeusmp2 point-to-point communication time

- Transitions at timesteps 41 and 69 clearly visible
- High variation among processes
- Progressive increase
  - from 0.4 to 1.2 in the maximum values
- Some processes, like processes 10 and 13 have low values where others are much higher
132.zeusmp2 late sender time

- Around half of the point-to-point time is spent in late sender situations
- Same kind of transitions and progressive increase present
132.zeusmp2 collective communication time

- Graph at same scale as the graphs on the point-to-point slides
- Peaks visible at iterations 41 and 69, where the transitions happen on the point-to-point graphs
132.zeusmp2 bytes transferred

- There is a variation among processes, but
- There is no variation over time
- The visit count graphs look exactly the same
- The progressive increase and the transitions in point-to-point time have to be caused by something else!
132.zeusmp2 late sender time

- After identifying the most interesting iterations
- Back to the SCALASCA GUI
- Look at the details
- Find the cause of the performance bottlenecks
SPEC MPI 2007 32-way execution time graphs
SPEC MPI 2007 32-way execution time charts
SPEC MPI 2007 32-way point-to-point time graphs
SPEC MPI 2007 32-way point-to-point time charts
Conclusion

- SCALASCA analysis was possible for all the SPEC MPI 2007 applications, up to 1024 processes
- Phase instrumentation adds a new dimension to the existing SCALASCA analysis
- Analyzing the SPEC MPI 2007 suite, we have found interesting time-dependent behavioral patterns
Thank you!

For more information visit our website:

http://www.scalasca.org

WRF-NMM weather prediction code on MareNostrum @ 1600 CPUs
Creating filters

- A filter file is a simple list of function names
- The SCALASCA toolset helps in choosing the right functions to filter

<table>
<thead>
<tr>
<th>flt</th>
<th>type</th>
<th>max_tbc</th>
<th>time</th>
<th>% region</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANY</td>
<td>2458949120</td>
<td>82441.94</td>
<td>100.00</td>
<td>(summary) ALL</td>
</tr>
<tr>
<td>MPI</td>
<td>9597728</td>
<td>18926.53</td>
<td>22.96</td>
<td>(summary) MPI</td>
</tr>
<tr>
<td>USR</td>
<td>2447099328</td>
<td>59827.53</td>
<td>72.57</td>
<td>(summary) USR</td>
</tr>
<tr>
<td>COM</td>
<td>2261016</td>
<td>3687.87</td>
<td>4.47</td>
<td>(summary) COM</td>
</tr>
<tr>
<td>USR</td>
<td>406944000</td>
<td>2471.16</td>
<td>3.00</td>
<td>__tf_projection_NMOD_totalisation_advection</td>
</tr>
<tr>
<td>USR</td>
<td>406944000</td>
<td>3244.68</td>
<td>3.94</td>
<td>__tf_projection_NMOD_interpolation_ordre_4</td>
</tr>
<tr>
<td>USR</td>
<td>406944000</td>
<td>4717.67</td>
<td>5.72</td>
<td>__tf_1d_hydro_NMOD_tf_1d_hydro_driver</td>
</tr>
<tr>
<td>USR</td>
<td>406944000</td>
<td>12794.70</td>
<td>15.52</td>
<td>__tf_projection_NMOD_tf_projection_mono</td>
</tr>
<tr>
<td>USR</td>
<td>406944000</td>
<td>8112.65</td>
<td>9.84</td>
<td>__tf_1d_hydro_NMOD_schema_de_godunov</td>
</tr>
<tr>
<td>USR</td>
<td>406944000</td>
<td>28249.64</td>
<td>34.27</td>
<td>__tf_thermo_NMOD_exact_riemann_solver</td>
</tr>
<tr>
<td>USR</td>
<td>5425920</td>
<td>226.38</td>
<td>0.27</td>
<td>__tf_ad_splitting_NMOD_tf_apply_boundary_conditions</td>
</tr>
<tr>
<td>MPI</td>
<td>3962400</td>
<td>12808.47</td>
<td>15.54</td>
<td>MPI_Send</td>
</tr>
<tr>
<td>MPI</td>
<td>3639888</td>
<td>9.52</td>
<td>0.01</td>
<td>MPI_Irecv</td>
</tr>
<tr>
<td>MPI</td>
<td>1899072</td>
<td>2651.80</td>
<td>3.22</td>
<td>MPI_Wait</td>
</tr>
<tr>
<td>COM</td>
<td>1085184</td>
<td>4.70</td>
<td>0.01</td>
<td>__tf_ad_splitting_NMOD_tf_communications_plane_init</td>
</tr>
</tbody>
</table>

...     ...     ...     ...     ...
Filter generation

- Example filtering 4 functions of 129.tera_tf:

<table>
<thead>
<tr>
<th>flt</th>
<th>type</th>
<th>max_tbc</th>
<th>time</th>
<th>%</th>
<th>region</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>FLT</td>
<td>1627776000</td>
<td>42078.13</td>
<td>51.04</td>
<td>(summary) FLT</td>
</tr>
<tr>
<td>*</td>
<td>ANY</td>
<td>831173120</td>
<td>40363.81</td>
<td>48.96</td>
<td>(summary) ALL-FLT</td>
</tr>
<tr>
<td>-</td>
<td>MPI</td>
<td>9597728</td>
<td>18926.53</td>
<td>22.96</td>
<td>(summary) MPI-FLT</td>
</tr>
<tr>
<td>*</td>
<td>USR</td>
<td>819323328</td>
<td>17749.41</td>
<td>21.53</td>
<td>(summary) USR-FLT</td>
</tr>
<tr>
<td>*</td>
<td>COM</td>
<td>2261016</td>
<td>3687.87</td>
<td>4.47</td>
<td>(summary) COM-FLT</td>
</tr>
</tbody>
</table>

- Example filtering 6 functions of 129.tera_tf:

<table>
<thead>
<tr>
<th>flt</th>
<th>type</th>
<th>max_tbc</th>
<th>time</th>
<th>%</th>
<th>region</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>FLT</td>
<td>2441664000</td>
<td>59590.50</td>
<td>72.28</td>
<td>(summary) FLT</td>
</tr>
<tr>
<td>*</td>
<td>ANY</td>
<td>17285120</td>
<td>22851.44</td>
<td>27.72</td>
<td>(summary) ALL-FLT</td>
</tr>
<tr>
<td>-</td>
<td>MPI</td>
<td>9597728</td>
<td>18926.53</td>
<td>22.96</td>
<td>(summary) MPI-FLT</td>
</tr>
<tr>
<td>*</td>
<td>USR</td>
<td>5435328</td>
<td>237.03</td>
<td>0.29</td>
<td>(summary) USR-FLT</td>
</tr>
<tr>
<td>*</td>
<td>COM</td>
<td>2261016</td>
<td>3687.87</td>
<td>4.47</td>
<td>(summary) COM-FLT</td>
</tr>
</tbody>
</table>
107.leslie3d execution time

- every 5\textsuperscript{th} iteration takes more time
  - caused by an additional MPI collective call
- around every 43\textsuperscript{rd} iteration takes longer
  - caused by longer point-to-point communication times
  - number of point-to-point calls and bytes transferred are the same
  - possibly caused by flooded communication buffers
- transition at iteration number 1015
113. GemsFDTD execution time

- some variation among processes visible
- slight transitions at a number of iterations
- some noise present
115.fds4 execution time

- the 1\textsuperscript{st} iteration takes longer
- every 3\textsuperscript{rd} iteration spends more time in user functions
- point-to-point communication only every 100\textsuperscript{th} iteration
- more point-to-point and collective calls in the first 100 iterations
- more collective calls every 100\textsuperscript{th} iteration
- more collective calls every 255\textsuperscript{th} iteration
121.pop2 execution time

- 2000 iteration experiment
  - *Remainder of 9000 iterations qualitatively similar*
- slight transitions in collective and point-to-point call counts
- more collective calls every 100th iteration
126. lammps execution time

- progressively increasing baseline time
- more point-to-point calls every 20\textsuperscript{th} iteration
- collective communication every 100\textsuperscript{th} iteration
127.wrf2 execution time

- slightly increasing baseline
- some periodic behavior, most visible in the peaks
- 1\textsuperscript{st} and every 1200\textsuperscript{th} iterations extremely long (every simulation day)
- more time spent in user functions every 4\textsuperscript{th} and 8\textsuperscript{th} iteration
- collective communication every 300\textsuperscript{th} iteration
128.GAPgeofem execution time

- 1\textsuperscript{st} iteration takes longer
  - visible in MPI time, but
  - MPI call count and bytes transferred is the same
- decreasing visit numbers and execution times
- Collectives every iteration
- Considerable amount of noise
  - mostly from collectives
  - Most of collective time spent in Wait at NxN situations
  - Shows load imbalance
129.tera_tf execution time

- Strong increase in execution time from 1.2s to 2.9s
- High amount of noise
- Visit numbers and bytes transferred
  - vary by processes, but
  - constant through iterations
- Strange, constantly changing behavior on communication time metrics
- Point-to-point and collective times complement each other
130. socorro execution time

- low number of iterations
- very long time spent in each iteration
- last iteration takes less time
137.lu execution time

- Collective communication only in last iteration
- Execution time and communication metrics mostly constant
- Transition around the 19\textsuperscript{th} iteration
- Variation among processes in communication time and count metrics
132.zeusmp2 collective communication time

- 5.6% of total time spent in collective communication
- of which 59.6% is spent on one callpath in an MPI_Allreduce
- Shows 47.5% standard variation
- The processes in the middle spend less time here than the ones on the edges
132.zeusmp2 wait at NxN time

- 3.9% of total time spent in wait at NxN situations
- looking at the MPI_Allreduce call in the $1^{st}$ iteration shows
  - 35.7% standard variation
  - highest waiting times on the bottom planes and
  - in the first few rows of every plane
  - lowest waiting times in the middle
Trace pattern example: Late sender

- Waiting time caused by a blocking receive operation posted earlier than the corresponding send operation
- Applies to blocking as well as non-blocking communication