Models

• Model
  • physical, conceptual, or mathematical **representation of a real** phenomenon that is difficult to observe directly
  • a **simplified and idealized** understanding of physical systems

• Used to:
  • **Explain, understand, insight**
    • Current behavior
    • Steer refactoring
    • Co-design runtime
  • Control
  • Predict

A model is as good as the data you use to build it
A model is as good as the use you make of it
BSC Tools framework

Trace handling & display
Simulators
Analytics

Open Source
http://www.bsc.es/paraver

Extrae
MRNET
LD_PRELOAD, Valgrind, Dyninst, PIN PAPI

Instr. traces
.dim
.prv
Time Analysis, filters
.prv + .pcf

Paraver
Performance analytics

Instr. traces
.dim
.prv2dim

DIMEMAS
VENUS (IBM-ZRL)

Machine description

Instruction level simulators
Tasksim, Sniper, gems

Performance analytics

XML control

Machine description

The importance of detail and flexibility
Since ~ 1991
Modeling in our framework ...

• Trace manipulation
  • Filter, cut, shift, ...

• Performance Analytics to detect structure & grain
  • Clustering
  • Tracking
  • Folding
  • spectral

• Modeling tools
  • Dimemas
  • Performance/CPIstack, Power
  • Hardware counters projection
  • Efficiency model
  • Performance extrapolation

• The really important thing:
  • “holistic” integration of data manipulation, analytics and models
  • Towards insight
Clustering to identify structure

Object Tracking

1. Capture a video (sequence of frames).
2. Identify objects of interest within each frame (e.g. contour detection).
3. Correlate the objects between pairs of frames.
   • Rules: characteristics specific to the objects and the physical system (e.g. color, speed)
Tracking structural evolution

- Frame sequence: clustered scatterplot as core counts increases

Dimemas: Coarse grain, Trace driven simulation

- Simulation: Highly non linear model
  - Linear components
    - Point to point communication
    - Sequential processor performance
      - Global CPU speed
      - Per block/subroutine
  - Non linear components
    - Synchronization semantics
      - Blocking receives
      - Rendezvous
      - Eager limit
    - Resource contention
      - CPU
      - Communication subsystem
        - links (half/full duplex), busses

$$T = \frac{MessageSize}{BW} + L$$

Fast & Fundamentals
What ifs ...

What if …

... we increase the IPC of Cluster1?

... we balance Clusters 1 & 2?
Multiscale Simulation

• Combine use of analytics and simulator
  • Steer what to simulate to reduce total cost

• Combination of trace driven core model, execution driven runtime

“Simulating Whole Supercomputer Applications”. Gonzále, J, et all. IEEE MICRO May 2011

CPI breakdown models

• Analytics: enabler !!

• Vendor models

“CPI analysis on POWER5, Parts 1 & 2”
IBM Systems & Technology Group Systems Performance.
(Re)using - (re)factoring old techniques

- Energy aware runtime
  - Dynamic periodicity detection (DPD)
  - Performance and Power model

- Dynamic frequency setting policies
  - Minimize one metric with a threshold on other
    - energy to solution
    - time to solution

- Based on periodic structure detection


Lenovo - BSC
Simple model fitting

- Constant
- Linear

OpenMX

AMG2013
OpenIFS @ Thunder-X1

- Strong scaling tracking of IPC
  - Sharing effect: L2, BW, ...

- Core Cost Model
  - Ideal CPI (1.6), L2 hit cost (62.2), L2 miss cost (129,4)
  - “reverse” engineering form real production code runs

- Socket IPC: ~ 26 !!
OpenIFS @ Thunder-X1

• Fit IPC
  • Numerical ?
  • Behavioral ?
  • Interpretation ?
Application characterization

Efficiencies: \( \sim (0,1] \)
Multiplicative model

Other architectural effects

Global Efficiency

Computation Efficiency

Parallel Efficiency

IPC Efficiency

Frequency Efficiency

Instruction Efficiency

Load Balance

Communication Efficiency

Serialization Efficiency

Transfer Efficiency

Cache

Memory BW

NUMAness

Dependencies

Sharing effects

Instruction mix

Code replication

Synchronization

Scaling model

- Actual run
- Ideal machine: $L=0; BW=\infty$
- If no dependences

Also in Scalasca

- Load Balance
- Transfer
- Serialization
# Efficiencies

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parallel Efficiency</strong></td>
<td>0.9834</td>
<td>0.9436</td>
<td>0.8980</td>
<td>0.8478</td>
</tr>
<tr>
<td><strong>Load Balance</strong></td>
<td>0.9871</td>
<td>0.9687</td>
<td>0.9099</td>
<td>0.9177</td>
</tr>
<tr>
<td><strong>Serialization efficiency</strong></td>
<td>0.9975</td>
<td>0.9770</td>
<td>0.9938</td>
<td>0.9395</td>
</tr>
<tr>
<td><strong>Transfer Efficiency</strong></td>
<td>0.9988</td>
<td>0.9970</td>
<td>0.9931</td>
<td>0.9833</td>
</tr>
<tr>
<td><strong>Computation Efficiency</strong></td>
<td>1.000</td>
<td>0.9590</td>
<td>0.8680</td>
<td>0.6953</td>
</tr>
<tr>
<td><strong>Global efficiency</strong></td>
<td>0.9834</td>
<td>0.9049</td>
<td>0.7795</td>
<td>0.5894</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IPC Scaling Efficiency</strong></td>
<td>1.000</td>
<td>0.9932</td>
<td>0.9591</td>
<td>0.8421</td>
</tr>
<tr>
<td><strong>Instruction Scaling Efficiency</strong></td>
<td>1.000</td>
<td>0.9721</td>
<td>0.9393</td>
<td>0.9075</td>
</tr>
<tr>
<td><strong>Core frequency efficiency</strong></td>
<td>1.000</td>
<td>0.9932</td>
<td>0.9635</td>
<td>0.9098</td>
</tr>
</tbody>
</table>
### On performance portability/predictability

- Analyses from different training courses
  - Running lulesh on default system setup

<table>
<thead>
<tr>
<th>Code</th>
<th>Parallel efficiency</th>
<th>Communication efficiency</th>
<th>Load Balance efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>lulesh@machine1</td>
<td>90.55</td>
<td><strong>99.22</strong></td>
<td>91.26</td>
</tr>
<tr>
<td>lulesh@machine2</td>
<td><strong>69.15</strong></td>
<td>99.12</td>
<td><strong>69.76</strong></td>
</tr>
<tr>
<td>lulesh@machine3</td>
<td>70.55</td>
<td>96.56</td>
<td>73.06</td>
</tr>
<tr>
<td>lulesh@machine4</td>
<td>83.68</td>
<td>95.48</td>
<td>87.64</td>
</tr>
<tr>
<td>lulesh@machine5</td>
<td><strong>90.92</strong></td>
<td>98.59</td>
<td><strong>92.20</strong></td>
</tr>
<tr>
<td>lulesh@machine6</td>
<td>73.96</td>
<td>97.56</td>
<td>75.81</td>
</tr>
<tr>
<td>lulesh@machine7</td>
<td>75.48</td>
<td><strong>88.84</strong></td>
<td>84.06</td>
</tr>
<tr>
<td>lulesh@machine8</td>
<td>77.28</td>
<td>92.33</td>
<td>83.70</td>
</tr>
<tr>
<td>lulesh@machine9</td>
<td>88.20</td>
<td>98.45</td>
<td>89.57</td>
</tr>
<tr>
<td>lulesh@machine10</td>
<td>81.26</td>
<td>91.58</td>
<td>88.73</td>
</tr>
</tbody>
</table>

Huge variability. How will the next machine behave?

J. Gimenez, “it is me, or it is the machine?”. Scalable Tools Workshop 2017.
Powerful analysis ...

- Identify relevance of different factors ...
- ... trends and outliers
- In conjunction with detailed analysis of individual points

AVBP @ Jupora

IPC variability

MPI progression engine issue

AMG2013 @ MareNostrum
Extrapolations ...

Several core counts

eff_factors.py

eff.csv

extrapolation.py

"Scalability prediction for fundamental performance factors" J. Labarta et al. SuperFRI 2014

Intel – BSC Exascale Lab
Extrapolations ...

• Amdahl fit ...

\[ Amdahl_{fit} = \frac{metric_0}{f_{metric} + (1 - f_{metric}) \times P} \]

• ...reasonable in the absence of more detailed behavioral information
Powerful what ifs ...

Can eliminate CPU noise

prv2dim

Several core counts

Dimemas

Eliminate network/MPI noise
Model hypothetical target platforms

Can eliminate CPU noise

effFactors.py

eff.csv

extrapolation.py

No MPI noise

Intel – BSC
Exascale Lab

“Scalability prediction for fundamental performance factors”
J. Labarta et al.
SuperFRI 2014

+ No preemption noise
About Analytics AND infrastructure

PyCOMPSs

```python
for name in list_traces:
    prv = name
    row = prv[-4:] + '_row'
    pcf = prv[-4:] + '_pcf'
    dim = os.path.basename(prv)[-4:] + '_dim'
    prv_ideal = 'D.' + os.path.basename(prv)
    pcf_ideal = 'D.' + os.path.basename(pcf)
    row_ideal = 'D.' + os.path.basename(row)
    np = get_num_prs_trace(prv)
    prv2dim(prv, pcf, row, dim)
    dimemas(pcf, row, dim, prv_ideal, pcf_ideal, row_ideal,
             head_file, tail_file, col4dim, dim_ideal_sim)
    ulb = paramedir(prv_ideal, pcf_ideal, row_ideal,
                     '2dp_mpi_stats.cfg')
    lb, trf = paramedir(prv, pcf, row, '2dp_mpi_stats.cfg')
    collect_per_trace.append((np, ulb, lb, trf))
from pycompss.api.api import compss_wait_on
collect_per_trace = compss_wait_on(collect_per_trace)
```

@task(trace_prv=FILE_IN, trace_pcm=FILE_IN, trace_row=FILE_IN, cfg=FILE_IN, returns=list)
def paramedir(trace_prv, trace_pcm, trace_row, cfg, trace_id_name):
    #Body of function
```
Performance analytics and modeling towards insight

• The importance of Integrated data handling, analytics and models

• Balance between first principles, pure statistical, black box ?
  • “Proper” choice of feature vector

• We use models much less than desirable !!!
  • → we actually use unquantified mental models
  • Training / Best practices issue

• A couple of “recommendations”
  • Performance tools: leverage methods from ALL areas & big data infrastructures
  • Runtimes: malleability and adaptability
Thank you

Jesus.labarta@bsc.es