Deconstructing PARSEC Scalability

Gabriel Southern and Jose Renau
Dept. of Computer Engineering, UC Santa Cruz
PARSEC Reported Speedup – 2011

Figure 4.1 from *Benchmarking Modern Multiprocessors*, Christian Bienia, PhD thesis, 2011.

Figure 2 from *Thread Reinforcer: Dynamically Determining Number of Threads via OS Level Monitoring*, Kishore Kumar Pusukuri, Rajiv Gupta, Laxmi N. Bhuyan, IISWC 2011.
Reported Speedup for Blackscholes

• Bienia
  • Blackscholes can achieve 14 times speedup on 16 core system

• Pusukuri
  • Blackscholes has maximum speedup of 5 times on a 24 core system

• Why are there different results for the same benchmark?
Reported Speedup for Blackscholes

Bienia • Pusukuri et al.
Reported Speedup for Blackcholes

- PARSEC has four different input sets
- Two ways to measure execution time
- Bienia
  - Simlarge input set
  - Region of Interest (ROI)
- Pusukuri
  - Native input set
  - Full execution
Blackscholes – Reported vs. Measured

Reported

Measured

Speedup vs. Core / Threads

Speedup vs. PARSEC minimum threads parameter

Bienia
Pusukuri et al.

Blackscholes – Reported vs. Measured

PARSEC minimum threads parameter
Reporting PARSEC parameters

Reported parameters for papers using PARSEC published in ISCA from 2010 – 2014

<table>
<thead>
<tr>
<th>Input Set</th>
<th>Full</th>
<th>ROI</th>
<th>Not Stated</th>
</tr>
</thead>
<tbody>
<tr>
<td>native</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>simlarge</td>
<td>0</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>simmedium</td>
<td>0</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>simsmall</td>
<td>0</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>not stated</td>
<td>1</td>
<td>1</td>
<td>20</td>
</tr>
</tbody>
</table>
Outline

• Motivation

• PARSEC background
  • Input sets
  • Region of Interest (ROI)

• Evaluate
  1. Does the ROI matter?
  2. Do different input sets produce similar results?
  3. What are the benchmark scalability trends?

• Conclusion
PARSEC Background

- Benchmark suite of emerging parallel workloads
- Developed at Princeton between 2005 and 2009
- Consists of 13 multithreaded applications
- Users select minimum number of threads
- Multiple input sets
  - 1 for *native* execution
  - 3 simulation input sets
    - simlarge
    - simmedium
    - simsmall
- Parallel portion is *Region of Interest* (ROI)
PARSEC Region of Interest (ROI)

Native Input Set

Initialization

Begin ROI

Parallel code

... 

Parallel code

End ROI

Finalization

Simulation Input Sets

Init

Parallel

... 

Parallel

End ROI

Fin
1) Does ROI Matter?

PARSEC FAQ claims

1. Users should only measure execution time of ROI when using simulation inputs
   - The simulation inputs have a significantly reduced parallel phase that exaggerates the proportions of the serial initialization and shutdown phase.

2. Serial phases don’t matter when using native inputs
   - [Native inputs] provide input data more similar to real program problem sizes so that the serial initialization and shutdown phases of the program don't matter . . .
ROI Percentage for Single Thread

![Bar chart showing ROI percentage for single thread across various applications.](chart.png)
ROI Doesn’t Matter (7 benchmarks)
ROI Matters for Simulation Inputs

![Bar chart showing ROI for different simulation inputs.](chart.png)

- **Native**
- **Simmedium**
- **Simlarge**
- **Simsmall**

Simulation Inputs:
- Bodytrack
- Facesim
- Fluidanimate

Percent ROI:
- 0
- 20
- 40
- 60
- 80
- 100
ROI Matters for All Input Sets

![Bar chart](chart.png)

- **Native**
- **Simmedium**
- **Simlarge**
- **Simsmall**

**Percent ROI**

- Blackscholes
- Canneal
- Raytrace
Canneal

![Graph showing speedup over PARSEC minimum threads parameter for Native Full and Native ROI](image-url)
2) Different Inputs Same Results?

- PARSEC FAQ claims:
  - *We already took advantage of all possible ways to minimize run time without distorting the characteristics too much when we created the simulation inputs.*

- Our results show large variations in scalability between different input sets and benchmarks

- No single consistent trend
Experiment Methodology (*Raytrace*)

![Graph showing speedup vs PARSEC minimum threads parameter]
Maximum Speedup

![Bar chart showing maximum speedup for various benchmarks and scenarios. The chart includes native-full, native-roi, simlarge-full, simlarge-roi, simmedium-full, simmedium-roi, simsmall-full, and simsmall-roi. The benchmarks include blackscholes, bodytrack, canneal, dedup, facesim, ferret, fluidanimate, freqmine, raytrace, streamcluster, swaptions, vips, x264, and geomean.]
Maximum Speedup ROI Only

native-roi  simlarge-roi  simmedium-roi  simsmall-roi

Speedup

blackscholes  bodytrack  canmeal  dedup  facesim  ferret  fluidanimate  freqmine  raytrace  streamcluster  swapoptions  vips  x264  geomean
Large Variation in Scalability

[Bar chart showing the speedup for various applications across different ROI sizes (native-roi, simlarge-roi, simmedium-roi, simsmall-roi).]
Low Scalability for Simulation Inputs

Speedup

native-roi  simlarge-roi  simmedium-roi  simsmall-roi

0  10  20  30  40  50

blackscholes  bodytrack  canneal  dealpix  facesim  ferret  flin annotate  free-mine  raytrace  streamcluster  swaptions  vips  x264  geomean
3. Scalability Trends

- Low Scalability
  - bodytrack
  - facesim
  - ferret (sim inputs)
  - freqmin (sim inputs)
  - vips (sim inputs)
  - x264

- High Scalability
  - blackscholes
  - canneal
  - ferret (native input)
  - freqmine (native input)
  - raytrace
  - vips (native)

- Other
  - dedup
  - streamcluster
  - swaptions
Scalability Trend – Freqmine

PARSEC minimum threads parameter

Speedup
Erratic Scalability – Dedup

![Graph showing speedup over PARSEC minimum threads parameter]
Erratic Scalability – Dedup (Simlarge, ROI)
Conclusion

• Summary
  1. ROI compared to full
     • No difference for 7 benchmarks
     • Large difference for simulation inputs for 3 benchmarks
     • Large difference for all inputs for 3 benchmarks
  2. Comparing input sets
     • Large variation in scalability for different input sets for 5 benchmarks
     • Low scalability for simulation inputs for 8 benchmarks
  3. Scalability Trends
     • Varied: high, low,

• Advice
  • Authors should report region modeled and inputs set used
  • Reviewers and readers should be aware of differences
Questions

• ❓❓❓
Backup
Experiment Methodology

- Evaluate impact of ROI and input set selection on PARSEC scalability using real systems
- Measure runtime of full execution and ROI for each input set
- Use 3 different multiprocessors systems
  - 8 logical processors
    - 1 x Intel Xeon E3-1275 v3 (4 core, 2-way SMT)
  - 32 logical processors
    - 2 x Intel Xeon E5-2689 (8 core, 2-way SMT)
  - 48 logical processors
    - 4 x AMD Opteron 6172 (12 core)
- Vary PARSEC minimum threads parameter from 1 to maximum number of logical processors
PARSEC Input Sets

- PARSEC has 4 different size input sets
  - 1 called *native* that is intended to approximate realistic application input
  - 3 simulation inputs (simlarge, simmedium, simsmall) that were created by scaling down the native input set

- Inputs were selected so that serial execution on a real machine would complete in
  - 15 minutes for *native*
  - 15 seconds for *simlarge*
  - 4 seconds for *simmedium*
  - 1 second for *simsmall*
PARSEC FAQ Simulation Inputs and ROI

• Q: I don't get good speedups for the PARSEC workloads on my machine. Why?

• A: You probably use the simulation inputs and measure the execution time of the whole program. You should only measure the execution time of the Region-of-Interest (ROI) instead. The simulation inputs have a significantly reduced parallel phase that exaggerates the proportions of the serial initialization and shutdown phase. This causes an artificial reduction of the achievable speedup due to Amdahl's Law. The inflation of the initialization and shutdown phase relative to the size of the parallel phase is a scaling artifact that should be compensated for.
PARSEC FAQ Native Inputs and ROI

• Q: I would like to measure speedups or characteristics using the entire execution time of the workloads, not just the Region-of-Interest. What can I do?

• A: You should use the native inputs for that. They provide input data more similar to real program problem sizes so that the serial initialization and shutdown phases of the program don't matter, at least as long as you don't run performance experiments with very large numbers of cores.
Blacksholes

![Graph showing speedup against PARSEC minimum threads parameter across different scenarios.](image-url)
Bodytrack

![Graph showing speedup vs PARSEC minimum threads parameter]

- Native Full
- Simlarge Full
- Simmedium Full
- Simsmall Full
- Native ROI
- Simlarge ROI
- Simmedium ROI
- Simsmall ROI

Speedup vs PARSEC minimum threads parameter
Canneal

![Graph showing speedup vs PARSEC minimum threads parameter for different configurations.](image-url)
Dedup

![Graph showing speedup vs PARSEC minimum threads parameter]

- Native Full
- Simlarge Full
- Simmedium Full
- Simsmall Full
- Native ROI
- Simlarge ROI
- Simmedium ROI
- Simsmall ROI

Speedup vs PARSEC minimum threads parameter
Facesim

![Graph showing speedup vs PARSEC minimum threads parameter]

- Native Full
- Simlarge Full
- Simmedium Full
- Simsmall Full
- Native ROI
- Simlarge ROI
- Simmedium ROI
- Simsmall ROI

The graph illustrates the speedup of different configurations as the PARSEC minimum threads parameter varies. The x-axis represents the parameter, while the y-axis shows the speedup.
Ferret

![Graph showing speedup vs. PARSEC minimum threads parameter]
Freqmine
Raytrace

![Graph showing speedup vs. PARSEC minimum threads parameter]

- Native Full
- Simlarge Full
- Simmedium Full
- Simsmall Full
- Native ROI
- Simlarge ROI
- Simmedium ROI
- Simsmall ROI

WDDD 2015
UC SANTA CRUZ
Micro Architecture Santa Cruz
Gabriel Southern and Jose Renau
Deconstructing PARSEC Scalability
Streamcluster

![Graph showing Streamcluster performance across different PARSEC minimum threads parameters]
Swaptions

![Graph showing speedup vs. PARSEC minimum threads parameter]

- Native Full
- Simlarge Full
- Simmedium Full
- Simsmall Full
- Native ROI
- Simlarge ROI
- Simmedium ROI
- Simsmall ROI
Vips

Speedup

PARSEC minimum threads parameter

Native Full  
Simlarge Full  
Simmedium Full  
Simsmall Full  
Native ROI  
Simlarge ROI  
Simmedium ROI  
Simsmall ROI
x264

![Graph showing speedup vs. PARSEC minimum threads parameter]

- Native Full
- Simlarge Full
- Simmedium Full
- Simsmall Full
- Native ROI
- Simlarge ROI
- Simmedium ROI
- Simsmall ROI

**Speedup**

**PARSEC minimum threads parameter**