The ParLab Stack

Parallel Computing Laboratory
Sarah Bird
May 30, 2013
Easy to write correct programs that run efficiently on manycore
# Integrated Software Stack

## Applications

- Personal Health
- Image Retrieval
- Hearing, Music
- Speech
- Parallel Browser

## Motifs/Dwarfs

## Composition & Coordination Language (C&CL)

## C&CL Compiler/Interpreter

- Parallel Libraries
- Parallel Frameworks

## Efficiency Languages

- Legacy Code
- Schedulers
- Sketching
- Autotuners
- Communication & Synch. Primitives

## Efficiency Language Compilers

## OS

- Legacy OS
- OS Libraries & Services
- Hypervisor

## Arch.

- Multicore/GPGPU
- RAMP Manycore

## Correctness

- Static Verification
- Type Systems
- Directed Testing
- Dynamic Checking
- Debugging with Replay
Hardware

- RAMP Gold
  - Simulation
  - 64 cores
  - FPGA

- RISP-V
  - Implementations written in Chisel
  - Rocket
    - 6 stage in-order
  - Hwacha
    - 64 bit vector core
  - FPGA
    - 2 Rocket
  - 45 nm Chip
    - 1 Rocket, 1 Hwacha
    - 1 Ghz

- x86

- GPGPU
  - Cuda
  - OpenGL
Operating Systems

• Akaros
  – Cloud OS

• Tessellation
  – Client OS
  – Space-Time Partitioning
  – Two-Level Scheduling
  – QoS to Applications
  – PACORA

• Linux
Schedulers

- **Lithe**
  - Compose Parallel Runtimes
  - Thread Building Blocks
  - Open MP

- **PULSE**
  - Framework to write schedulers
  - Earliest Deadline-First
  - Global Round Robin

Diagram:

- Lithe
  - OpenMP
  - TBB
  - Lithe
  - Akaros

- PULSE
  - EDF
  - GRR
  - PULSE
  - Tessellation
  - Linux

- Platform options:
  - SPARC
  - RISC-V
  - x86
  - PTX
  - RAMP Gold
  - FPGA
  - Chip
  - Multicore
  - GPGPU
Why Create an Integrated Prototype?

- Encourages Collaboration
- Prevents neglecting important pieces of the problem
- Uncover opportunities for invention by seeing which side of an interface is the best place to satisfy a requirement
- Demonstrate the importance of design simplicity
- Enhance the education of the PhD students in areas beyond their own specialties
- Help with technology transfer by giving concrete examples of our ideas for our colleagues in industry
Forces for Integration

• Design Compatibility
  – Shared Space and Discussions
  – Symbiotic Designs
  – Example: Music and Tessellation

• Customized Support
  – In-house experts helping adapt their design to your problem
  – Examples: Lithe and Tessellation

CAA and Applications

• Motivating Applications
  – Exciting to show your research on run a compelling application
  – Examples: Patterns and MRI

BFS and RISC-V
## Integrated Demos in ParLab History

<table>
<thead>
<tr>
<th>Year</th>
<th>Month</th>
<th>Demo Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>January</td>
<td>1, 2, 3, 4, 5, 6, 7, 8, 9</td>
</tr>
<tr>
<td>2011</td>
<td>January</td>
<td>10, 11, 12, 13, 14, 15, 16, 17, 18, 19</td>
</tr>
<tr>
<td>2012</td>
<td>January</td>
<td>20, 21, 22, 23, 24, 25, 26, 27</td>
</tr>
<tr>
<td>2013</td>
<td>January</td>
<td>28, 29</td>
</tr>
</tbody>
</table>

Note: The table represents the demo numbers scheduled for each month from January 2010 to January 2013.
Stack Redesign

What did rethinking the entire computing stack at once get us?

• Productivity programs can create applications that require efficiency
  – Scale
  – Performance

• Easily target many platforms and features
  – Example: Vector units on RISC-V Chip

• Efficient performance predictibility
  – Interactivity and Responsiveness
  – Realtime Performance
Integrated Demos

• Two Demos to show off the ParLab Stack
• Fun and compelling applications that require efficiency
• Easily target many platforms and features
• Interactivity and Realtime Performance
• Integration!
Music Exploration and Recommendation

- Better Pandora
- Audio Content Analysis Framework
- Parallel Browser Big Data Visualization
- Demonstrates Scale and Responsiveness
Music Recommendation and Exploration

PARDORA -> Parallel Browser

PyCASP -> ASP

TBB

Lithe

Tessellation

RISC-V

x86

PV FPGA

Multicore

PTX

GPGPU

RISC-V

x86

RV FPGA

Multicore

GPGPU
Virtual Musical Instrument

- Musical Instrument using a camera
- Music Synthesis
- Vision Applications
- Demonstrate Realtime
Virtual Musical Instrument
Music Recommendation & Exploration Demo

Leo Meyerovich, Katya Gonina, Gage Eads, Eric Roman, Eric Battenberg, Henry Cook, Gerald Friedland

End of ParLab Celebration
May 30, 2013
Demo
System Overview

Client

- "Radiohead"
- "King Tubby"
- sid_1, sid_2

Layout Engine

1K clustered songs

Server

Recommendation Engine

1M songs

Parallel Browser

SEJITS

Tesselation
Client Architecture

Safari Web Browser

Parallel JavaScript Libraries

- Render
- Layout
- Parse (JSON)

- 1K clustered songs
- 1M songs

Synthesizer

offline: compile-time

viz.ftl
Music Recommendation Stack

PARDORA

PyCASP

ASP

CUDA / Cilk+

TBB

Lithe

Linux

Tessellation

PTX

x86

GPGPU

Multicore

x86

RISC-V

RV FPGA

x86

Multicore

OpenCL

GPGPU

x86

JavaScript
Offline Phase

Online Phase

Server Architecture

1M songs

SEJITS

Get potential neighbors using Collaborative Filtering

"Radiohead"

Tesselation

GPU

CPU

FPGA
Music Recommendation Stack

- **PARDORA**
  - **PyCASP**
    - **ASP**
      - **CUDA / Cilk+**

- **TBB**
  - **Lithe**
    - **Tessellation**
      - **PTX**
      - **x86**
      - **RISC-V**
      - **RV FPGA**
      - **Multicore**

- **Parallel Browser**
  - **OpenCL**
  - **GPGPU**
  - **Multicore**
  - **x86**
  - **Multicore**