VIRTUALIZATION OF HETEROGENEOUS MACHINES

David F. Bacon
WHAT HETEROGENEOUS MACHINES?

It’s

The Multicore Era!
For years, computer architects have been saying that a **big new idea** in computing was needed. Indeed, as transistors have continued to shrink, **rather than continuing to innovate**, computer designers have **simply adopted a so-called “multicore” approach**, where multiple processors are added as more chip real estate became available.
MEANWHILE...

GPU

FPGA

Tilera 64

Cell BE

PowerEN
THE MARCH TO HETEROGENEITY

AMD Fusion

NVIDIA Project Denver

Intel Stellaron
Should We Virtualize Heterogeneous Architectures?
How?
At the Language or System Level?
Or Both?
What is “Virtualization” Anyway?
THE ONLY 2 IDEAS IN COMPUTER SCIENCE

Hashing

Indirection

\[ f(x) \]
OVERLAP: SYNERGY AND CONFLICT

Memory

Threads

User-Mode ISA

Supervisor-Mode ISA
Language VMs for Heterogenous Systems
Heterogeneous Programming

Java
C++
Python

Cuda
OpenCL

C++Intrinsics

VHDL
Verilog
SystemC

CPU Compiler
GPU Compiler
Node Compiler
Synthesis

binary
binary
binary
bitfile

CPU
GPU
WSP
FPGA
THE LIQUID METAL PROGRAMMING LANGUAGE

Lime

Lime Compiler

CPU Backend
- bytecode
- CPU

GPU Backend
- binary
- GPU

Node Backend
- binary
- WSP

Verilog Backend
- bitfile
- FPGA
THE ARTIFACT STORE & EXCLUSION

Lime

Lime Compiler

bytecode

binary

binary

bitfile

Artifact Store

CPU

GPU

WSP

FPGA
Execution, Communication, and Replacement
THE LIME LANGUAGE:
VIRTUALIZING HETEROGENEOUS COMPUTATION
LIME: JAVA IS (ALMOST) A SUBSET

% javac MyClass.java
% java MyClass

% mv MyClass.java MyClass.lime
% limec MyClass.lime
% java MyClass

INCREMENTALLY USE LIME FEATURES
Java + What?

• Programmable Primitives
• Stream Programming
• Map & Reduce Operations

• Ways to describe static shape and size
STREAMING COMPUTATION
PIPELINE PARALLELISM
local char work(char c) {
    return toUpper(c);
}

var.uppercaser = task work;

Task Creation

Isolation Keywords

Task Creation

primitive filter

stream

stream

port type

port type

char

char

worker method

TASK CREATION (STATELESS)
var pipeline = task worker1 => task worker2 => task worker3;
**Sources and Sinks**

- **Source:** /tmp/mydata
- **Sink:** /dev/tty
- **Filter:**
  - reader(...) { ... }
  - worker(...) { ... }
  - writer(...) { ... }

**Heap**

**File System**
public static void main(string[][] args) {

    char[][] msg = {
        'H', 'E', 'L', 'L', 'O', ',', ',', ',',
        'W', 'O', 'R', 'L', 'D', ',', '!', ',', '
    };

    var hello = msg.source(1) =>
        task Character.toLowerCase(char) =>
        task System.out.print(char);

    hello.finish();
}
DEMO
HELLO WORLD
LIME/ECLIPSE ENVIRONMENT
package helloworld;

public class HelloWorld3b {

    public static void main(String[] args) {
        char[] msg = {'H', 'E', 'L', 'L', 'O', ',', ',', '
        'W', 'O', 'R', 'L', 'D', ',', '!', ',', '
        'n'};

        var hello = msg.source(1) =>
        task Character.toLowerCase(char) =>
        task System.out.print(char);

        hello.rendezvous();
    }
}
Stateful Tasks

var averager = task Averager().avg;

double avg(double x) {
    total += x;
    return total/++count;
}
```java
int work1(int i) {
    int r = i + x;
    x = i / 2;
    return { r, i };
}

int work2(int i) {
    return i * 3;
}
```

```
var matchedpipe = task AddStuff().work1 => # => task work2;
```
DEMO

N-BODY SIMULATION: CPU VS. GPU
9x SPEEDUP (9.26 GFLOPS) ON LAPTOP
VIRTUALIZATION OF DATA MOVEMENT
MAP & REDUCE OPERATIONS

DATA PARALLELISM
Array Parallelism

float[ ] a = ...;
float[ ] b = ...;

float[ ] c = a @+ b;

float sum = + ! c;
public local V get(K key) {
    Node[LINKS] row = nodes[hash(key)];
    boolean[LINKS] selections = row @ compareKey(key);
    V[LINKS] vals = row @ getValueOrDefault(selections);
    return ! ! vals;
}
VIRTUALIZATION OF DATA PARALLELISM
CURRENT RESULTS
HOW DO WE EVALUATE PERFORMANCE?

• Speedup for Naïve Users
  • How much faster than Java?

• Slowdown for Expert Users?
  • How much slower than hand-tuned low-level code?

• Our methodology:
  • Write/tune/compare 4 versions of each benchmark:
    • Java, Lime, OpenCL, Verilog
  • Doesn’t address flops/watt, flops/watt/$, productivity
**Expert vs Naïve Speedup: Kernel Time**

(Java Baseline)

<table>
<thead>
<tr>
<th>Implementation</th>
<th>GPU</th>
<th>FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>photomosaic</td>
<td>37x</td>
<td>150x</td>
</tr>
<tr>
<td>n-body</td>
<td>193x</td>
<td>92x</td>
</tr>
<tr>
<td>idct</td>
<td>10x</td>
<td></td>
</tr>
<tr>
<td>des</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Should We Virtualize Heterogeneous Architectures?
Yes We Can!
**SUMMARY**

- Heterogeneity is Here to Stay
- Multicore Isn’t Dead - Just Less Important
- HLLs Can Insulate Programmer from Low-Level Details
  - But not fundamental computational structure
- System-Level Virtualization is an Open (Hard) Problem
Questions?