System Software for Hybrid HPC Architectures

Deepika H.V
Hybrid Computing Group
C-DAC, Bangalore
Hybrid Architectures
Computing Shift to Accelerators

- Accelerator “cards” for standard cluster nodes (PCIe, or on-board HyperTransport/QPI: CRAY)
  - many (~50...500) “lightweight” cores (~ 1 GHz)
- High thread concurrency, fast (local) memories
- Programming paradigms:
  - use CPU for program control, communication and maximum single-thread performance
  - “offload” data-parallel parts of computation to accelerator for maximum throughput performance
  - Adapt to multiple programming & load-balancing, careful assessment of “speedups”
C-DAC Hybrid Cluster

System Configuration:
- Xeon E5 series
- PCIe 3.0
- GPU (Tesla K20)
- FPGA (PICO M503)
- IB Cluster

<table>
<thead>
<tr>
<th>Core Complexity</th>
<th>Multi Core</th>
<th>Many Cores</th>
<th>Customizable Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Efficiency</td>
<td>Low 1 GF/Watt</td>
<td>Medium 5 GF/Watt</td>
<td>High 14 GF/Watt</td>
</tr>
<tr>
<td>Throughput Orientation</td>
<td>Low</td>
<td>High</td>
<td>Flexible</td>
</tr>
</tbody>
</table>
Hybrid HPC System Software Stack

Applications
- Bioinformatics
- Cryptography
- Weather Modeling
- CFD

Program Development Environment
- IDE
- Compiler Extension
- App Library
- Debugger
- Profiler
- Parallelizing Polyhedral Compiler

Scheduler & Runtime
- Scheduler
- Runtime
- Data Management
- Dynamic Adaptive Scheduler for Irregular Applications

Base Software
- OS
- File System
- Device Drivers
- Hardware Libraries

Hardware
- CPU
- GPU
- FPGA
- ACC
- N/W
- Storage

- CDAC Components
- Collaboration Components
- Third Party Software
- Hardware
System Software Components Developed

- Hybrid IDE
- Hybrid Scheduler
- Hybrid Monitoring & Management
- OpenCL Generator
- Parallelizing polyhedral compiler
- Dynamic Adaptive scheduler for Irregular applications
- Hybrid Runtime
- Hybrid Debugger Study
- Domain Specific IP Cores/Libraries on accelerators
- Hybrid Application
  - Selective CATionic Antibacterial Peptides (SCAAP)
**Integrated Development Environment**

- Hybrid IDE aims to provide an integrated environment with all the tools required to simplify writing hybrid parallel programs on heterogeneous clusters.
I want Biryani
I want Idli
I want Potato Curry
I want Mixed Veg
I want Soyabean

OK!, So I have to cook 5 dishes. Get the order of dishes sorted on priority & resources req. Fetch the suitable resources to cook them efficiently.

2 microwaves can cook the first 4 dishes in them very quickly. But to cook the 5th dish in microwave minimum wait-time is 25 mins. So why not use the available cooker saving 5 minutes of time.

Wow! Now that’s intelligence!!

<table>
<thead>
<tr>
<th>Item</th>
<th>Biryani</th>
<th>Idli</th>
<th>Potato Curry</th>
<th>Mixed Veg</th>
<th>Soyabean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooking Time - Microwave</td>
<td>15</td>
<td>15</td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>Cooking Time - Cooker</td>
<td>45</td>
<td>30</td>
<td>30</td>
<td>35</td>
<td>40</td>
</tr>
</tbody>
</table>
Hybrid Cluster Scheduler cont’d

Features

• Scheduling across heterogeneous resources
• Efficient scheduling algorithm
• Application Aware Scheduling
• Better turn-around time for jobs
• Improved resource utilization
• Dynamic fault tolerance
• Load balancing
• Integrated into Torque 4.2.8 as a policy
Hybrid Monitoring

The Hybrid monitoring tool collects, monitors, and helps to visualize & analyse the metrics such as CPU load, memory availability etc. for hybrid multi accelerator cluster systems to ensure normal functioning of the cluster, detecting service degradations and prompt rectification.

- Compute – load, availability, temperature
- Network – bandwidth, RTT
- Storage – availability
- user jobs - no. of jobs, resources, jobs status, execution time
- services - status of services

This tool can be extendable to:

- Monitor any new accelerator / device
- Analysis of archived data
- Alert facility
- Auto correction for specific errors
- Configurator
OpenCLGen: Automatic Program Generator

OpenCLGen aims to automatically generate OpenCL program from the kernel code freeing the application developers from the complex steps of OpenCL coding.

OpenCLGen service takes as input the kernel code & kernel parameters and gives as output the complete OpenCL program.

Benefits
- Improves programmer productivity by freeing the programmer from writing complex OpenCL codes
- Can be accessed from anywhere over Internet
The aim of this work is to provide an environment to permit users to use multiple heterogeneous devices (FPGA, GPU, CPU)

- Insulation of hardware heterogeneity and complexities
- Job execution and control
- Load Balancing between CPU/GPU/FPGA
- Data Management among compute elements

We customized the StarPU runtime (StarPU runtime developed by INRIA provides unified view of heterogeneous computational resources and optimized data transfer)

C-DAC Enhanced Runtime Features
- Identify and execute on FPGA devices
- Dynamic Reconfiguration to support both OpenCL and CUDA on the same GPGPU
Domain Specific IP Cores/Libraries

FPGA IP Cores

- **Arithmetic Cores on FPGA**
  - Floating point arithmetic IP cores
  - 1024-point Fixed point FFT IP cores
  - Fixed point BLAS IP core

- **Crypto Cores for Popular Hashing Algorithms**
  - SHA-1, SHA-224, SHA-256, MD5

GPU Libraries

- **Open CL Crypto Library for GPU**
  - MD5, SHA-1, SHA-256, SHA-512, SHA384, AES
Selective CAtionic Antibacterial Peptides (SCAAP)

- Given a peptide sequence find out whether it is an antibacterial peptide or not (positively charged peptide = cationic peptide)
- Calculates:
  - Mean Hydrophobicity
  - Mean Net Charge
  - Iso Electric Point
  - Hydrophobic Moment
- If the above parameters fall within the range then it is antibacterial peptide

Implementation:
- The above functionality have been implemented as FPGA IP cores as well as GPU kernels
Monte Carlo PI Calculation

A simple Monte Carlo simulation to approximate the value of π could involve randomly selecting points \( \{(x_i, y_i)\}_{i=1}^n \) in the unit square and determining the ratio \( p = m/n \), where \( m \) is the number of points that satisfy \( x^2 + y^2 \leq 1 \).

![Diagram showing task division between GPU & FPGA](image)

<table>
<thead>
<tr>
<th>Compute Element</th>
<th>Hybrid</th>
<th>FPGA Virtex5</th>
<th>GPU Tesla K20</th>
<th>CPU Xeon E5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency (MHz)</td>
<td>FPGA:GPU 200 : 706</td>
<td>200</td>
<td>706</td>
<td>2600</td>
</tr>
<tr>
<td>Time (ms)</td>
<td>201</td>
<td>205</td>
<td>207</td>
<td>4446</td>
</tr>
<tr>
<td>Speedup</td>
<td>22x</td>
<td>21x</td>
<td>21x</td>
<td>1x</td>
</tr>
</tbody>
</table>
THANK YOU