The High Performance Internet of Things: using GVirtuS for gluing cloud computing and ubiquitous connected devices

G. Laccetti°, R. Montella*

C. Palmieri**, V. Pelliccia**

*Department of Science and Technologies – University of Napoli Parthenope
° Department of Mathematics and Applications – University of Napoli Federico II
#Developer Team members
Motivations

• This preliminary work is addressed to speculate about:
  – The next generation of “off the shelf Beowulf clusters”
  – How to accelerate the Internet of Things component: High Performance Internet of Things

“neoWulf”
Generic Virtualization Service
(since March 2010)

- Framework for split-driver based abstraction components
- Plug-in architecture

**Independent from:**
- Hypervisor (or no-hypervisor)
- Communication
- Target of virtualization
- Architecture!

**High performance:**
- Enabling transparent virtualization
- With overall performances better or not too far from un-virtualized resources

http://osl.uniparthenope.it/projects/gvirtus/
Generic Virtualization Service
(since March 2010)

- From Google Scholar...

A GPGPU transparent virtualization component for high performance computing clouds
G Giunta, R Montella, G Agrillo, G Coviello
Euro-Par 2010-Parallel Processing, 379-391

A GPU Accelerated High Performance Cloud Computing Infrastructure for Grid Computing Based Virtual Environmental Laboratory
G Giunta, R Montella, G Laccetti, F Isaila, JG Blas
Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th ...

Virtualizing general purpose GPUs for high performance cloud computing: an application to a fluid simulator
R Di Lauro, F Giannone, L Ambrosio, R Montella
Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th ...

A general-purpose virtualization service for HPC on cloud computing: an application to GPUs
R Montella, G Coviello, G Giunta, G Laccetti, F Isaila, JG Blas
Parallel Processing and Applied Mathematics, 740-749

SlaaS-Sensing Instrument as a Service Using Cloud Computing to Turn Physical Instrument into Ubiquitous Service
R Di Lauro, F Lucarelli, R Montella
Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th ...

http://osl.uniparthenope.it/projects/gvirtus/
Split-Driver approach

- **Split-Driver**
  - Hardware access by privileged domain.
  - Unprivileged domains access the device using a frontend/backend approach.

- **Frontend (FE):**
  - Guest-side software component.
  - Stub: redirect requests to the backend.

- **Backend (BE):**
  - Manage device requests.
  - Device multiplexing.
GVirtuS approach

- GVirtuS Frontend
  - Dynamic loadable library
  - Same application binary interface
  - Run on guest user space

- GVirtuS Backend
  - Server application
  - Run in host user space
  - Concurrent requests
The Communicator

- Provides a high performance communication between virtual machines and their hosts.
- The choice of the hypervisor deeply affects the efficiency of the communication.

<table>
<thead>
<tr>
<th>Hypervisor</th>
<th>FE/BE comm</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>No hypervisor</td>
<td>Unix Sockets</td>
<td>Used for testing purposes</td>
</tr>
<tr>
<td>Generic</td>
<td>TCP/IP</td>
<td>• Communication testing purposes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Remote / Distributed virtualized resources (i.e. GPUs)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• High Performance Internet of Things</td>
</tr>
<tr>
<td>Xen</td>
<td>XenLoop</td>
<td>• runs directly on the top of the hardware through a custom Linux kernel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• provides a communication library between guest and host machines</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• implements low latency and wide bandwidth TCP/IP and UDP connections</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• app transparent and offers an automatic discovery of the supported VMs</td>
</tr>
<tr>
<td>VMware</td>
<td>Virtual Machine Communication Interface (VMCI)</td>
<td>• commercial hypervisor running at the application level</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• provides a datagram API to exchange small messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• a shared memory API to share data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• an access control API to control which resources a virtual machine can access</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• and a discovery service for publishing and retrieving resources</td>
</tr>
<tr>
<td>KVM/QEMU</td>
<td>VMchannel</td>
<td>• Linux loadable kernel module now embedded as a standard component</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• supplies a high performance guest/host communication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• based on a shared memory approach</td>
</tr>
</tbody>
</table>
An application: Virtualizing GPUs

- GPUs
  - Hypervisor independent
  - Communicator independent
  - GPU independent

The host plus a collection of devices managed by the OpenCL framework that allow an application to share resources and execute kernels on devices in the platform.
```c
#include <stdio.h>
#include <CL/cl.h>

int main(void) {
    cl_int error, platforms;
    cl_platform_id platform;
    error = clGetPlatformIDs(1, &platform, &platforms);
    printf("Number of platforms GPU(s): %d\n", platforms);
    return 0;
}
```

```c
extern "C" CL_API_ENTRY cl_int CL_API_CALL clGetPlatformIDs(
    cl_uint num_entries,
    cl_platform_id *platforms,
    cl_uint *num_platforms){
    OpenclFrontend::Prepare();
    OpenclFrontend::AddVariableForArguments(num_entries);
    OpenclFrontend::AddVariableForArguments(platforms);
    OpenclFrontend::AddVariableForArguments(num_platforms);
    OpenclFrontend::Execute("clGetPlatformIDs");
    if(OpenclFrontend::Success()){
        cl_uint *tmp_num_platform;
        tmp_num_platform = OpenclFrontend::GetOutputHostPointer<cl_uint>();
        if (tmp_num_platform != NULL)
            *num_platforms = *tmp_num_platform;
        cl_platform_id *tmp_platform;
        tmp_platform = (OpenclFrontend::GetOutputHostPointer<cl_platform_id>());
        if (tmp_platform != NULL)
            *platforms = *tmp_platform;
    }
    return new Result(exit_code, out);
}
```
Accelerating ARM boards

• High Performance Computing will be ARM based
  – Cheaper and powerful
  – Low heat emission
  – High developable

• High Performance Internet of Things
  – Small and smart devices highly pervasive
Figuring out the next generation HPC

- **x86_64:**
  - Head node
  - Login nodes
  - I/O nodes
    - Computing nodes for the cluster
    - I/O nodes for sub clusters

- **ARM:**
  - Computing nodes as multicore sub clusters
Enhancing the next generation of [low cost/middle end] HPC

**GPU:**
- Acceleration nodes

**High End GPGPU devices**
- Nvidia
  - Tesla
  - Fermi
  - Kepler
- ATI Radeon

**Hierarchical parallelism:**
- DM among I/O Nodes
- SM in I/O Nodes
- DM among sub clusters nodes
- SM in sub cluster CN(multicore ARMs)
- OpenCL Kernels (in CN/Acc Nodes)

NB: In this scenario we have ARM boards with a low performance GPU useless for HPC jobs
Distributed GPUs

Highlights:

• Using the Tcp/Ip Communicator FE/BE could be on different machines.
• ARM machines can access remote GPUs.

Applications:

• GPU for embedded systems as network machines
• Next generation of High Performance [Cloud] Computing
Prototyping

• Computing node:
  – Raspberry Pi mod. B. rev. 2
  – Wheezy Raspbian Linux

• Acceleration:
  – Genesis GE-i940 Tesla
  – i7-940 2.93 133 GHz fsb, Quad Core hyper-threaded 8 Mb cache CPU and 12Gb RAM.
  – 1 nVIDIA Quadro FX5800 4Gb RAM video card
  – 2 nVIDIA Tesla C1060 4 Gb RAM

• I/O:
  – Intel Xeon quad core HT
  – Ubuntu Linux 64bit
Evaluation

- Single computing node
...from (NVIDIA) OpenCL SDK...

- **ScalarProd** computes $k$ scalar products of two real vectors of length $m$.
  - Notice that each product is executed by an OpenCL thread on the GPU.
  - No synchronization is required.

- **MatrixMul** computes a matrix multiplication.
  - The matrices are $m \times n$ and $n \times p$, respectively.
  - It partitions the input matrices in blocks and associates an OpenCL thread to each block.
  - No need of synchronization.

- **Histogram** returns the histogram of a set of $m$ uniformly distributed real random numbers in 64 bins.
  - The set is distributed among the OpenCL threads each computing a local histogram.
  - The final result is obtained through synchronization and reduction techniques.
Benchmarking: ARM11

Problem Size (MB)

Wall Clock (s)

MatrixMul

ScalarProd

Histogram

Poor ARM Guy! 😊
Benchmarking: GPU!

ARM11 OpenCL gVirtuS x86_64 NvidiaTesla

$10^1, 10^0$ (CPU) $\rightarrow 10^{-3}, 10^{-4}$ (GPU)
(computing + communication)

You like to win easily, don't you? 😊
Conclusions and Future Works

- A subset of OpenCL has been ported under ARM platform as GVirtuS stub library.

- Tests and benchmarks demonstrate the performance improvements.

LESSONS LEARNT:
- Technically, accelerate an ARM powered board using OpenCL is possible:
  - In a fully transparent way
  - Using hybrid environment ARM/x86-64/GPU/Acc (i.e. Xeon Phi)
- Pervasive/wearable computers and "things" over the internet could be accelerated in the same way.
- The pervasive/wearable computers and "things" could be instanced as cloud "virtual" machines.

The winning architecture for building exascale systems, heterogeneous or homogeneous, and why?

- Multicore: Maintain complex cores, and replicate (x86, SPARC, Power7) [#3, 6, and 10]

- Manycore/Embedded: Use many simpler, low power cores from embedded (BlueGene, future ARM) [ #2, 4, 5, and 9]

- GPU/Coprocessor/Accelerator: Use highly specialized processors from graphics market space (NVidia Fermi, Intel Xeon Phi, AMD) [# 1, 7, and 8]

Jack Dongarra
Warsaw, 2013 – 9/9th
Conclusions and Future Works

• More evaluation
• Multiple computing nodes
Conclusions and Future Works

• Repeat testing on better ARM board(s) i.e.
  – Quad Core CPU 1.2 GHz
  – 1 GB RAM
  – Giga Ethernet

• Full OpenCL porting

• Application framework for the ARM “sub cluster” support

• Development of a comfortable SDK for hybrid programming

GVirtuS is GPL3 licensed open source! Download it at http://osl.uniparthenope.it/projects/gvirtus/
Thank You!

http://osl.uniparthenope.it/projects/gvirtus/