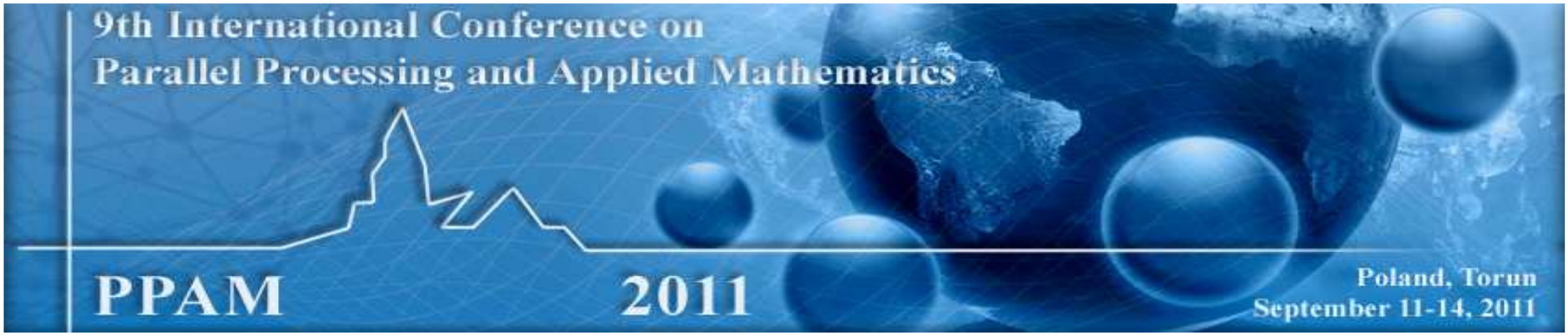


A general-purpose virtualization service for HPC on cloud computing: an application to GPUs

*R.Montella, G.Coviello, G.Giunta**
G. Laccetti#, F. Isaila, J. Garcia Blas°

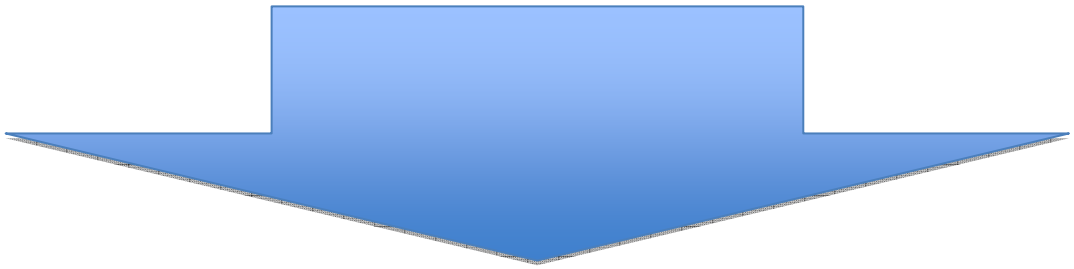
*Department of Applied Science – University of Napoli Parthenope
° Department of Mathematics and Applications – University of Napoli Federico II
°Department of Computer Science – University of Madrid Carlos III

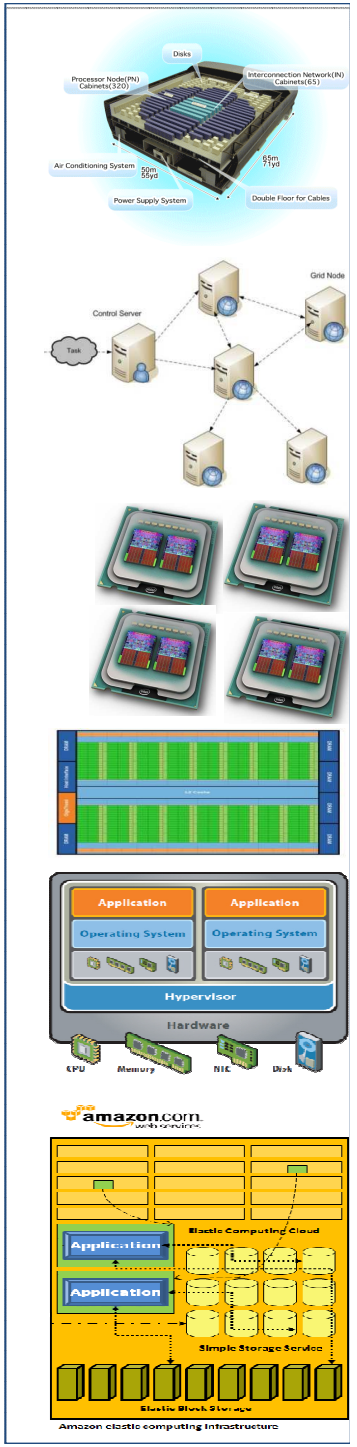
Outline

- Introduction and contextualization
- GVirtuS: Generic Virtualization Service
- GPU virtualization
- High performance cloud computing
- Who uses GVirtuS
- Conclusions and ongoing projects

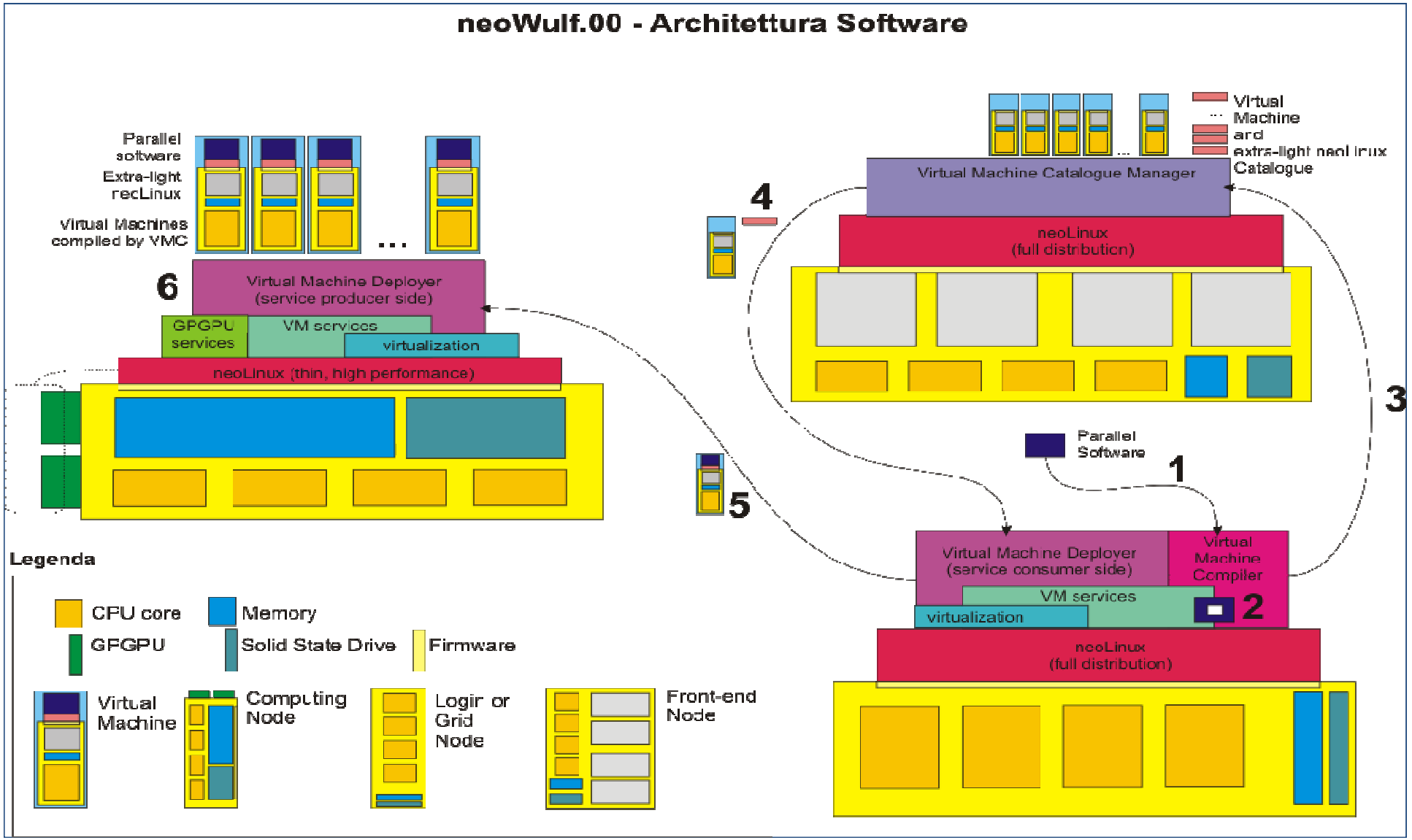
Introduction and contextualization

- High Performance Computing
- Grid computing
- Many core technology
- GPGPUs
- Virtualization
- Cloud computing





High Performance **Cloud** Computing



GVirtuS

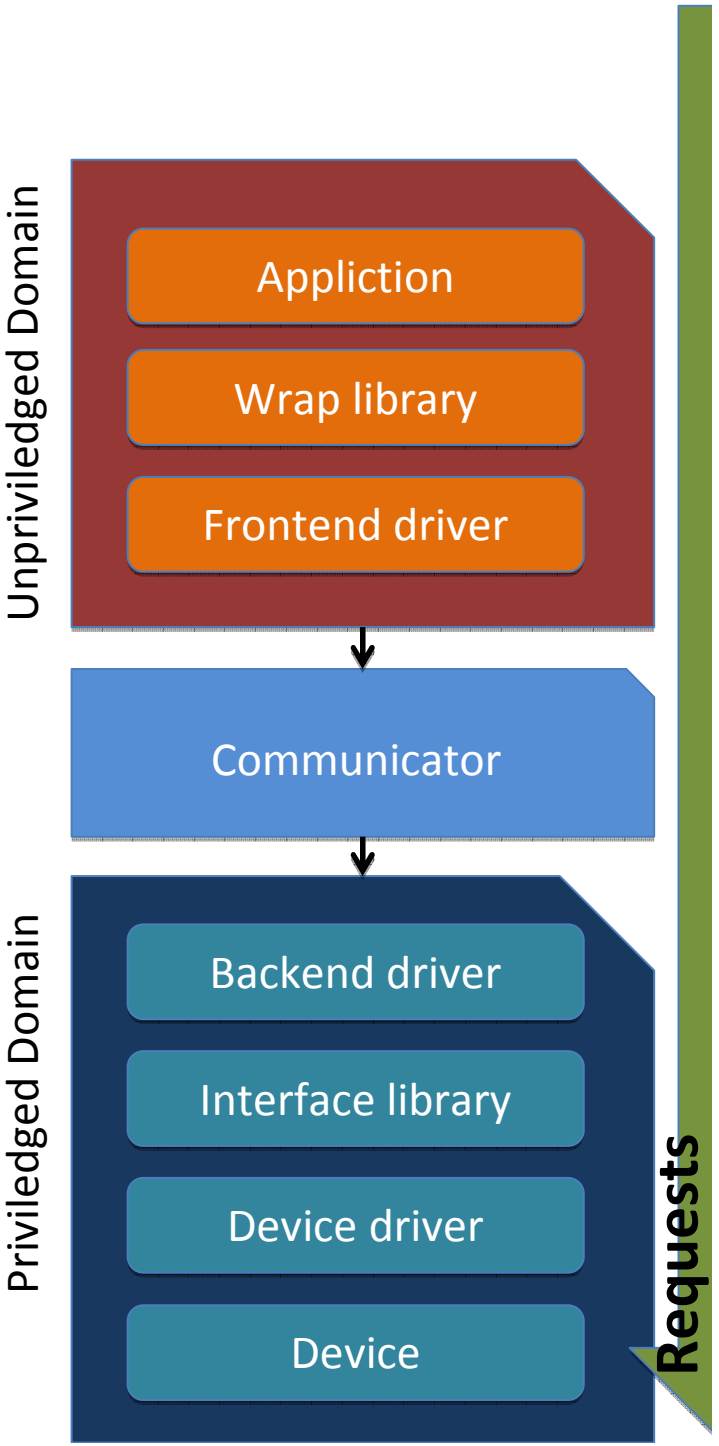
- Generic Virtualization Service
- Framework for split-driver based abstraction components
- Plug-in architecture

- Independent form
  - Hypervisor
  - Communication
  - Target of virtualization

- High performance:
  - Enabiling transparent virtualization
  - Wth overall performances not too far from un-virtualized machines

Split-Driver approach

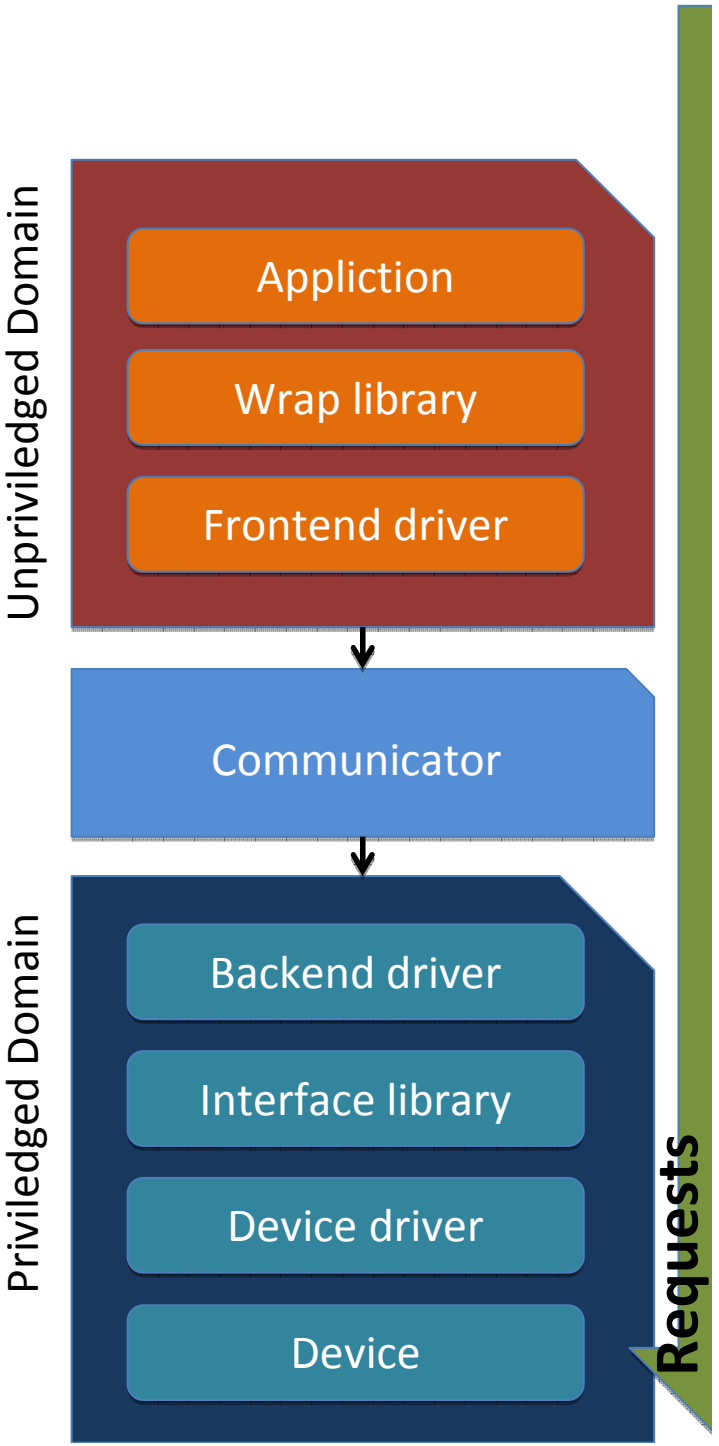
- Split-Driver
  - Hardware access by priviledged domain.
  - Unpriviledged domains access the device using a frontend/backhend approach
- Frontend (FE):
  - Guest-side software component.
  - Stub: redirect requests to the backend.
- Backend (BE):
  - Mange device requests.
  - Device multiplexing.



GVirtuS approach

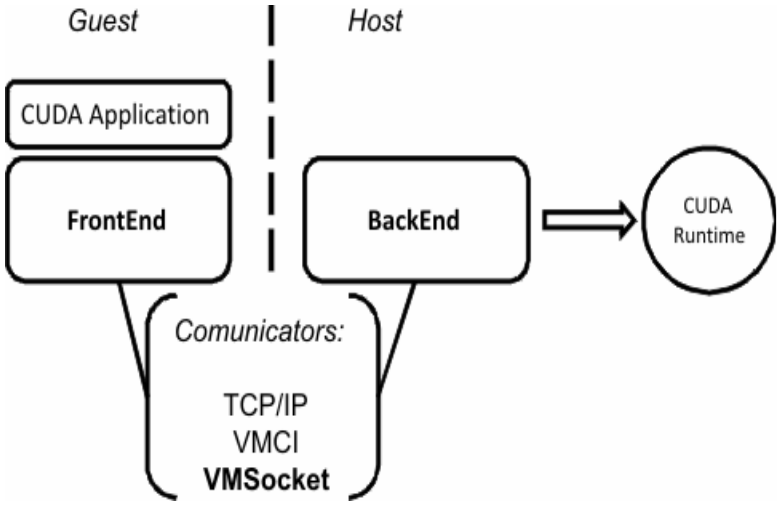
- GVirtuS Frontend
  - Dyinamic loadable library
  - Same application binary interface
  - Run on guest user space

- 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.



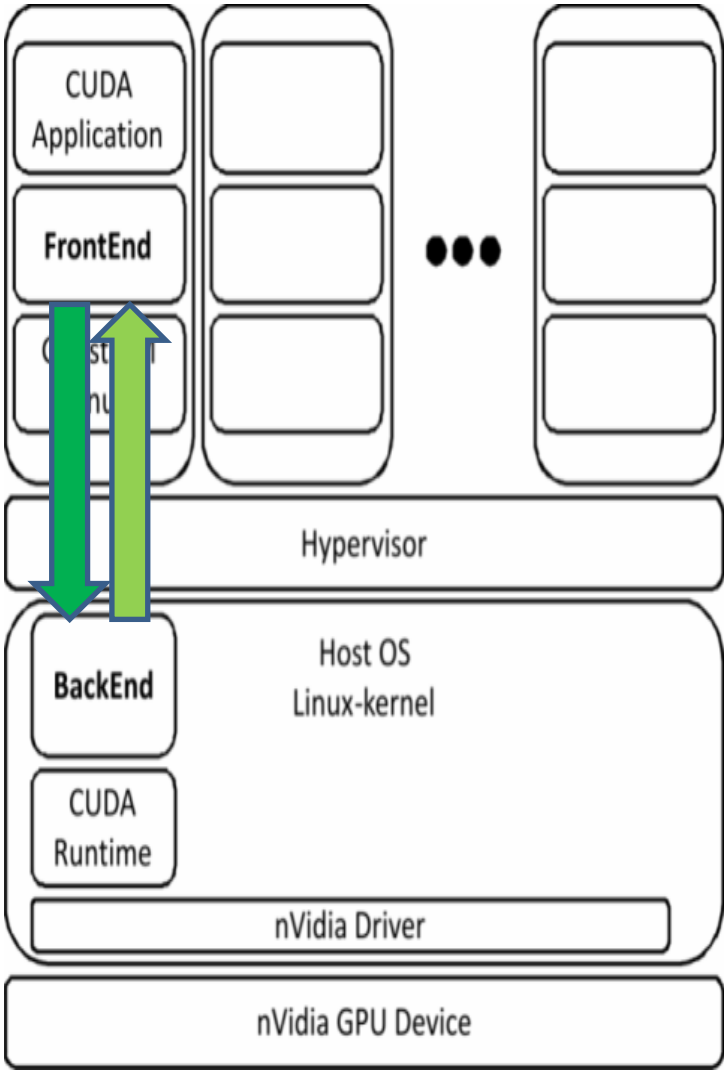
Hypervisor	FE/BE comm	Notes
No hypervisor	Unix Sockets	Used for testing purposes
Generic	TCP/IP	Used for communication testing purposes, but interesting…
Xen	XenLoop	•runs directly on the top of the hardware through a custom Linux kernel •provides a communication library between guest and host machines •implements low latency and wide bandwidth TCP/IP and UDP connections •application transparent and offers an automatic discovery of the supported VMS
VMware	Virtual Machine Communication Interface (VMCI)	•commercial hypervisor running at the application level. •provides a datagram API to exchange small messages •a shared memory API to share data, •an access control API to control which resources a virtual machine can access •and a discovery service for publishing and retrieving resources.
KVM/QEMU	VMchannel	•Linux loadable kernel module now embedded as a standard component •supplies a high performance guest/host communication

An application: Virtualizing GPUs

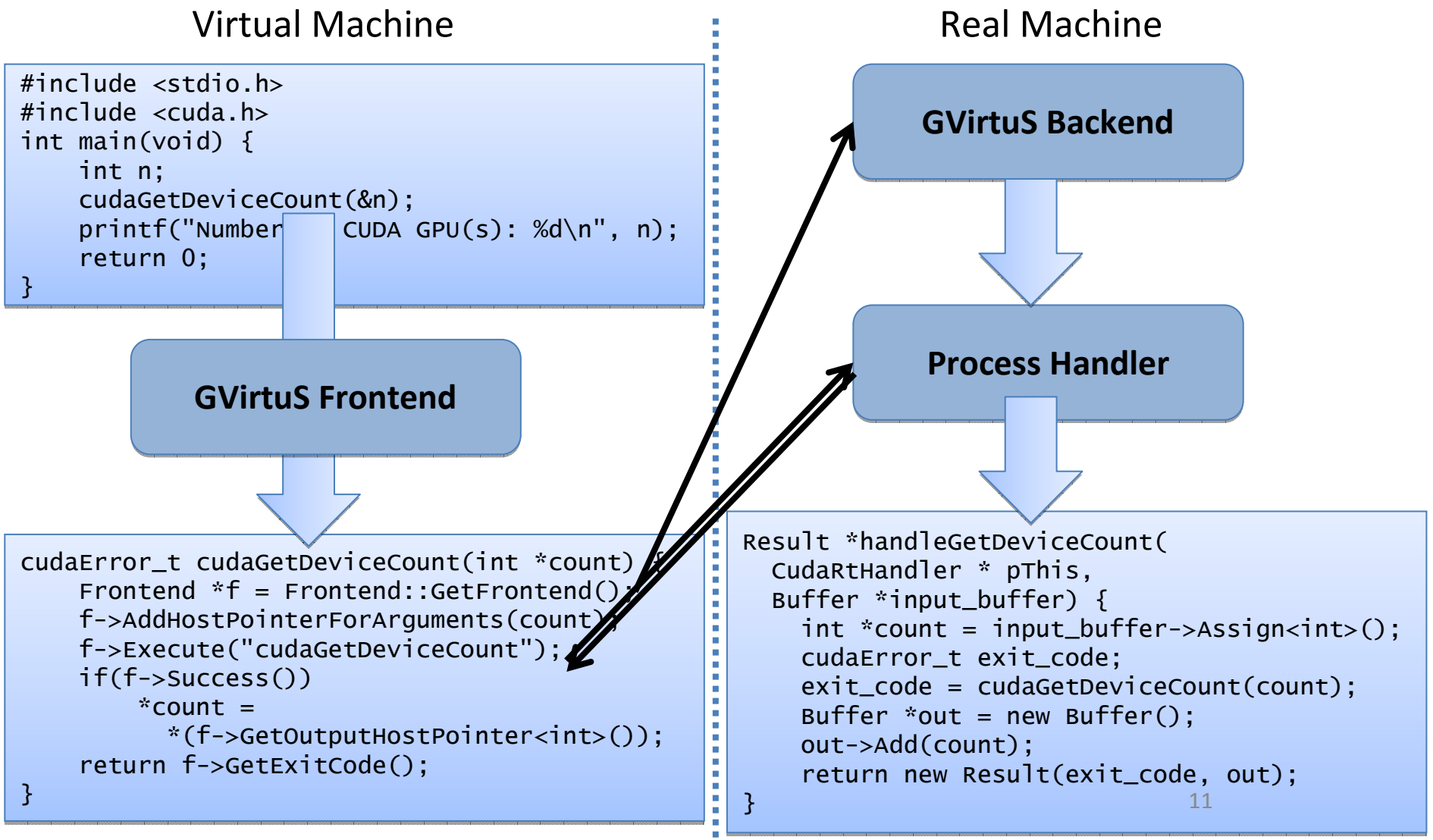
- **GPUs**
  - Hypervisor independent
  - Communicator independent
  - GPU independent

GVirtuS - CUDA

- Currently full threadsafe support to
  - CUDA drivers
  - CUDA runtime
  - OpenCL
- Partially supporting (it is needed more work)
  - OpenGL integration



GVirtuS – libcudard.so

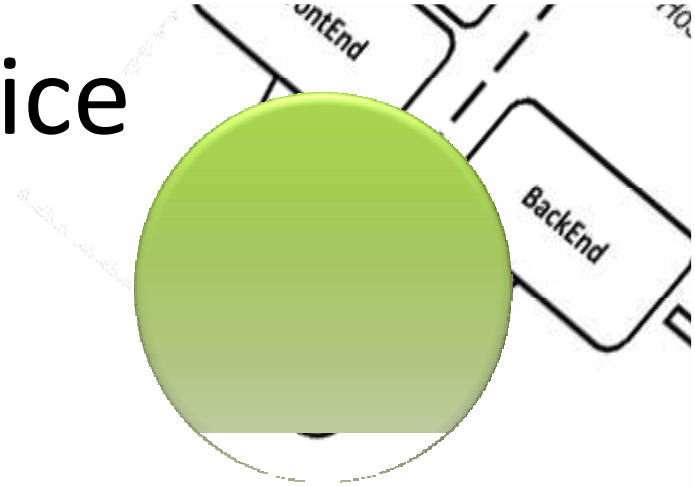


Choices and Motivations

Hypervisor	FE/BE Comm	Open Src	Running as	Official CUDA Drivers
~~Xen~~	~~Xen Loop~~	~~Yes~~	~~Kernel~~	~~No~~
~~VM Ware~~	~~VMCI~~	~~No~~	~~Application~~	~~Shares the host OS ones~~
KVM/QEMU	**vmSocket**	Yes	Loadable Kernel Module	Shares the host OS ones

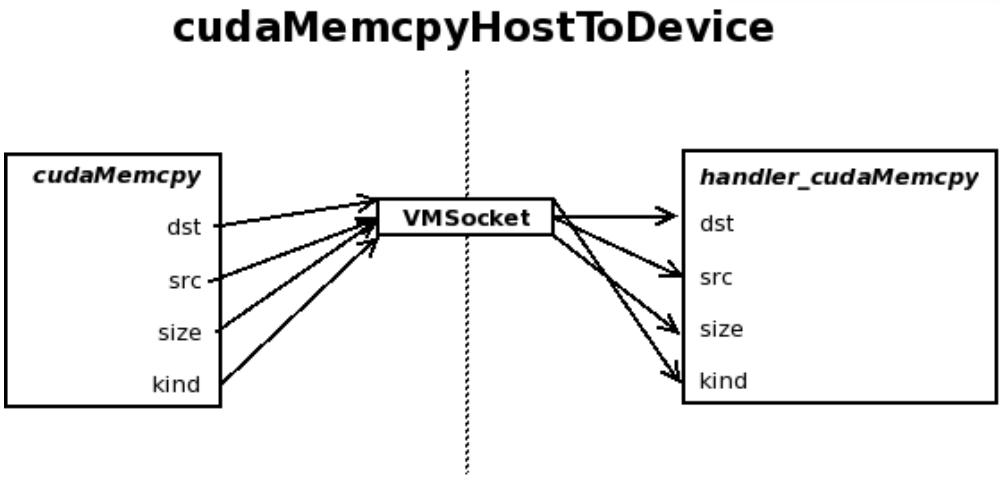
- **We focused on VMware and KVM hypervisors.**
- **vmSocket** is the component we have designed to obtain a high performance communicator
- **vmSocket** exposes Unix Sockets on virtual machine instances thanks to a QEMU device connected to the virtual PCI bus.

vmSocket: virtual PCI device



- Programming interface:
  - Unix Socket

- Communication between guest and host:
  - Virtual PCI interface
  - QEMU has been modified



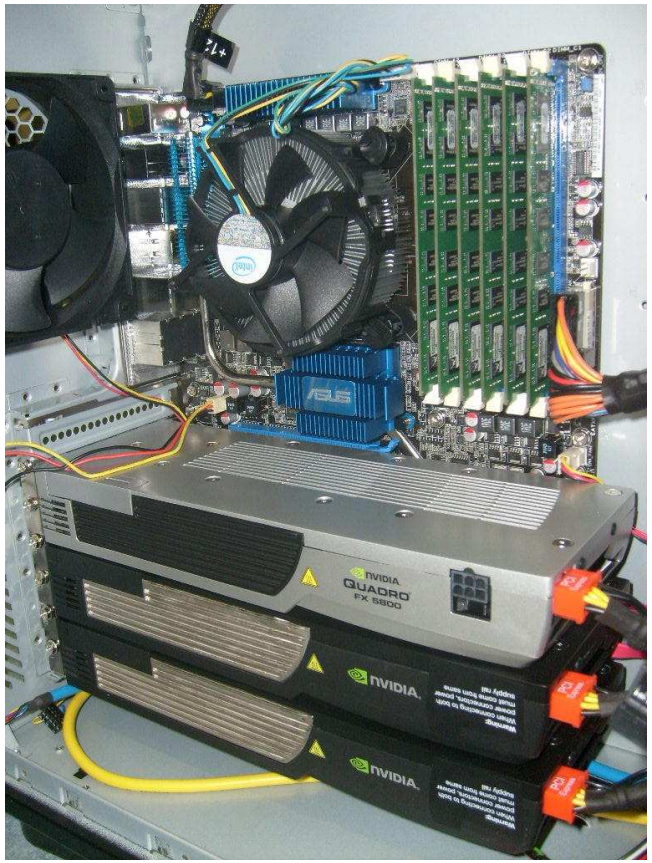
FE/BE interaction efficiency:

- there is no mapping between guest memory and device memory
- the memory device pointers are never de-referenced on the host side
- CUDA kernels are executed on the BE where the pointers are fully consistent.

Performance Evaluation

- CUDA Workstation
  - 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

- The testing system:
  - Ubuntu 10.04 Linux
  - nVIDIA CUDA Driver, and the SDK/Toolkit version 4.0.
  - VMware vs. KVM/QEMU (using different communicators).



GVirtuS-CUDA runtime performances

Results:

•0: No virtualization, no accleration (blank)
•1: Acceleration without virtualization (target)

•2,3: Virtualization with no acceleration

•4...6: GPU acceleraion, Tcp/Ip communication ⇒ Similar performances due to communication overhead

•7: GPU acceleration using GVirtuS, Unix Socket based communication

•8,9: GVirtuS virtualization

⇒Good performances, no so far from the target
⇒4..6 better performances than 0

Evaluation:
CUDA SDK benchmarks

#	Hypervisor	Comm.	Histogram	matrixMul	scalarProd
0	Host	**CPU**	100.00%	100.00%	100.00%
1	Host	**GPU**	9.50%	9.24%	8.37%
2	Kvm	**CPU**	105.57%	99.48%	106.75%
3	VM-Ware	**CPU**	103.63%	105.34%	106.58%
4	Host	**Tcp/Ip**	67.07%	52.73%	40.87%
5	Kvm	**Tcp/Ip**	67.54%	50.43%	42.95%
6	VM-Ware	**Tcp/Ip**	67.73%	50.37%	41.54%
7	Host	**AfUnix**	11.72%	16.73%	9.09%
8	Kvm	**vmSocket**	15.23%	31.21%	10.33%
9	VM-Ware	**vmcl**	28.38%	42.63%	18.03%

Computing times as Host-CPU rate

Distributed GPUs

Hilights:

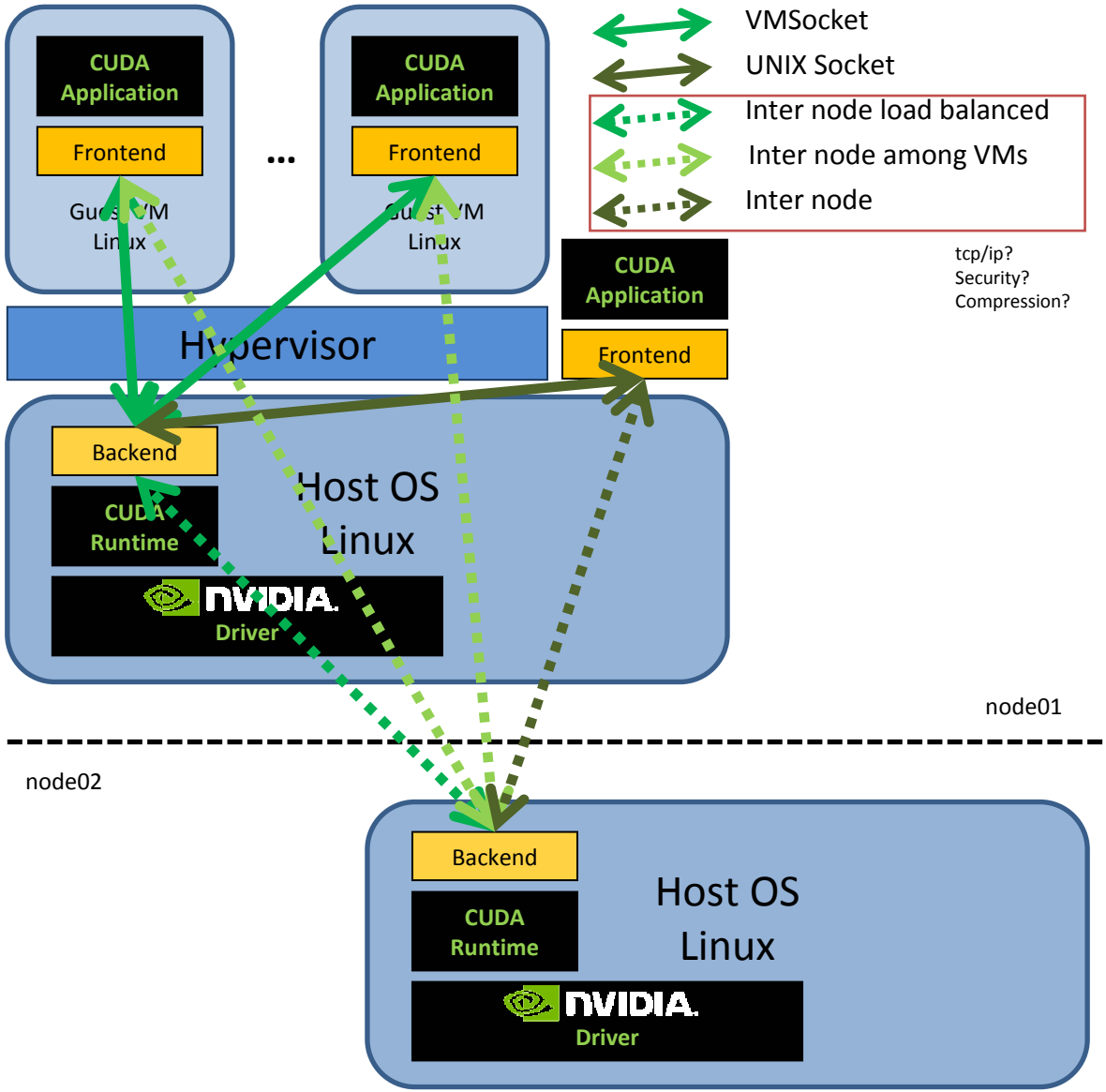
•Using theTcp/Ip Communicator FE/BE could be on different machines.

•Real machines can access remote GPUs.

Applications:

•GPU for embedded systems as network machines

•High Performance Cloud Computing

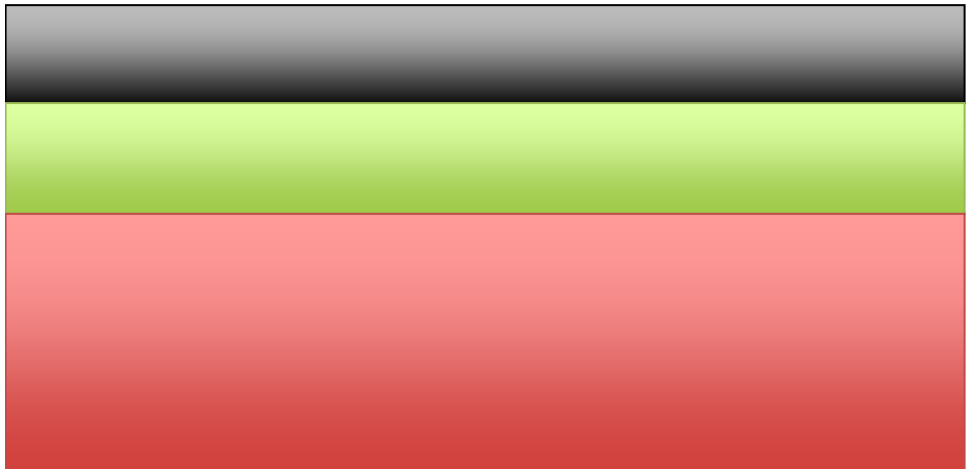


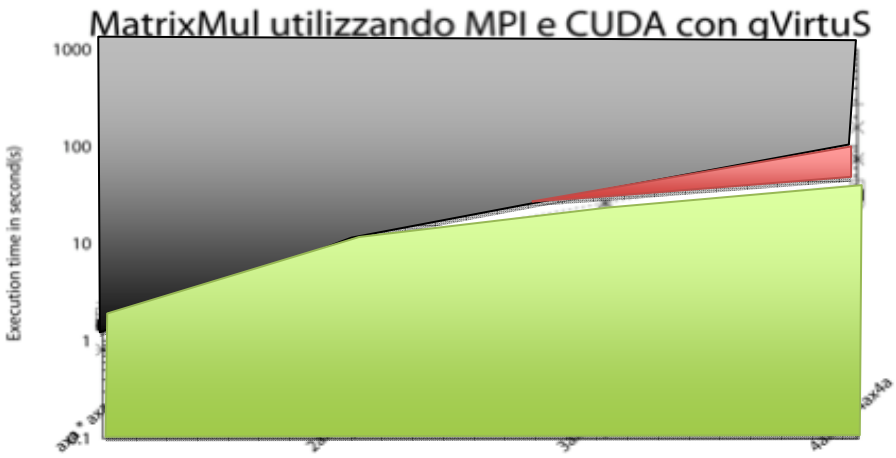
High Performance Cloud Computing

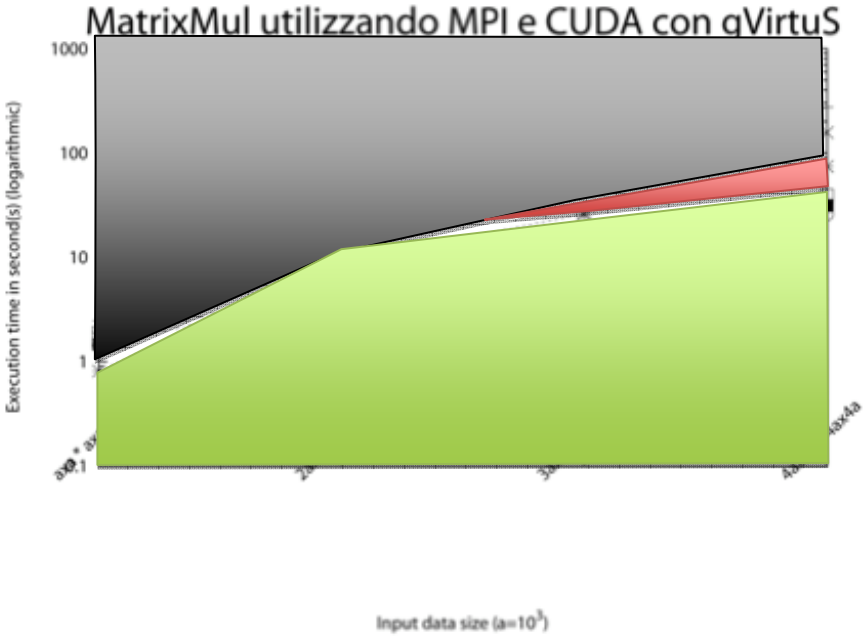
- *Ad hock* performance test for benchmarking
- Virtual cluster on local computing cloud

- **Benchmark:**
  - Matrix-matrix multiplication
  - 2 parallelims levels: distributed memory and GPU

- **Results:**







GVirtuS in the world

- GPU support to OpenStack cloud software
  - **Heterogeneous cloud computing**
    John Paul Walters et Al.
    University of Southern California / Information Sciences Institute
  - http://wiki.openstack.org/HeterogeneousGpuAcceleratorSupport
- HPC at NEC Labs of America in Princeton, University of Missouri Columbia, Ohio State University Columbus
  - **Supporting GPU Sharing in Cloud Environments with a Transparent Runtime Consolidation Framework**
  - Awarded as the best paper at HPDC2011

Download, *taste*, contribute!

- http://osl.uniparthenope.it/projects/gvirtus
  GPL/LGPL License

Conclusions

- The GVirtuS generic virtualization and sharing system enables thin Linux based virtual machines to use hosted devices as nVIDIA GPUs.
- The GVirtuS-CUDA stack permits to accelerate virtual machines with a small impact on overall performance respect to a pure host/gpu setup.
- GVirtuS can be easily extended to other enabled devices as high performance network devices

Ongoing projects

- Elastic Virtual Parallel File System
- MPI wrap for High Performance Cloud Computing
- XEN Support (is a big challenge!)

Download
Try & Contribute!