NETWORK-BASED COMPUTING LABORATORY

Virtual Machine Aware Communication Libraries for High Performance Computing

Wei Huang, Matthew Koop, Qi Gao, and Dhabaleswar K. Panda

Network Based Computing Laboratory
The Ohio State University

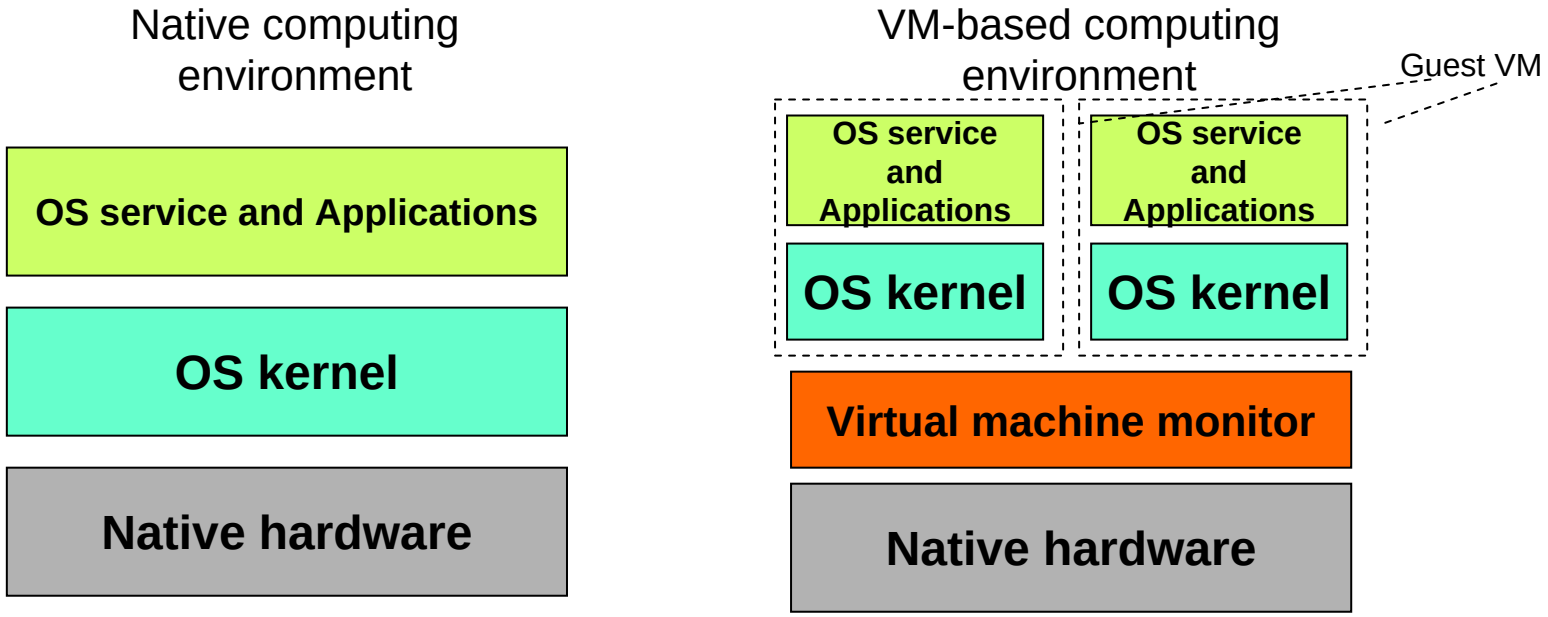
SC'07 -- Nov 13th, 2007

OHIO STATE

In this presentation …

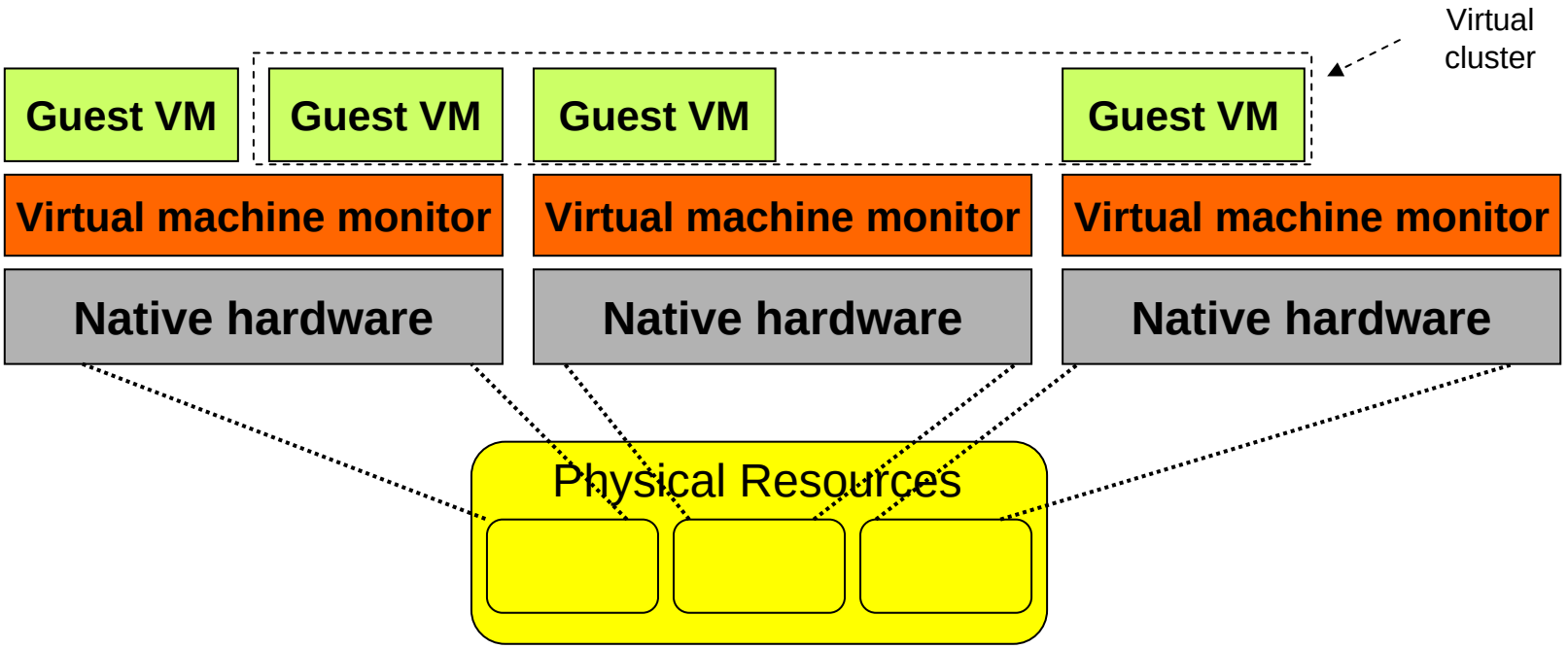
- We target high performance computing with virtual machines
- Why do we want to do this?
- What is missing?
  - Performance concerns
  - Efficient inter-VM communication
- What do we do?
  - IVC: Inter-VM Communication library
  - MPI: hiding the design complexities
  - Performance evaluation
- Conclusion

Virtual machine environment



- Virtual machine (VM) technologies allow running OSes on virtualized hardware instead of native hardware
- A wide adoption of VM environments:
  - Server consolidation: efficiently utilize the resources
  - Debugging and development: safety and efficiency

VM for HPC: how and why?

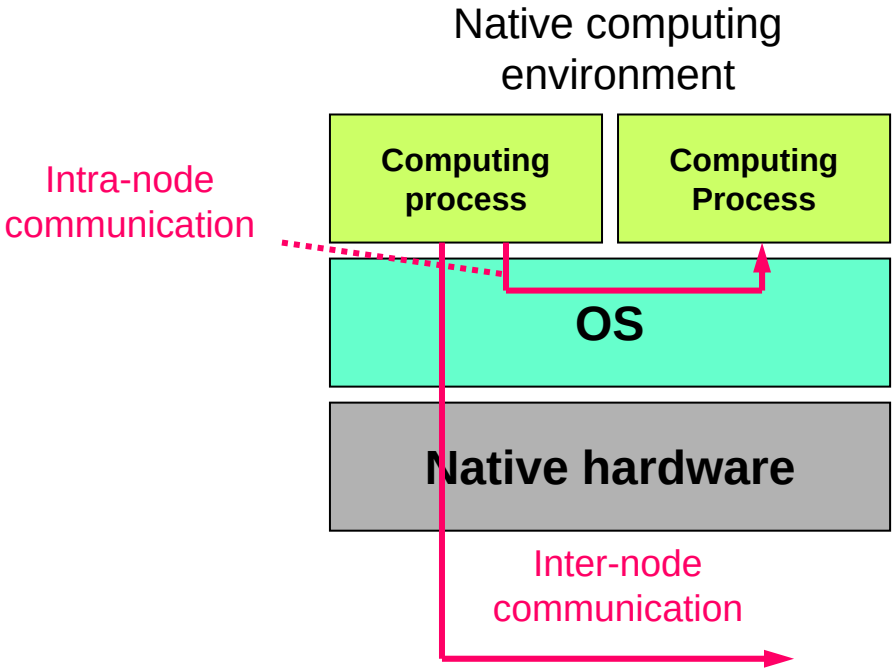


- Applications are running on virtual clusters consisting of multiple VMs
- VMs can be migrated among physical hosts
- Why VM based environments?
  - Management: hardware maintenance …
  - Fault tolerance
  - And many others: customized OSes, load balancing, performance isolation …

VM for HPC: why not?

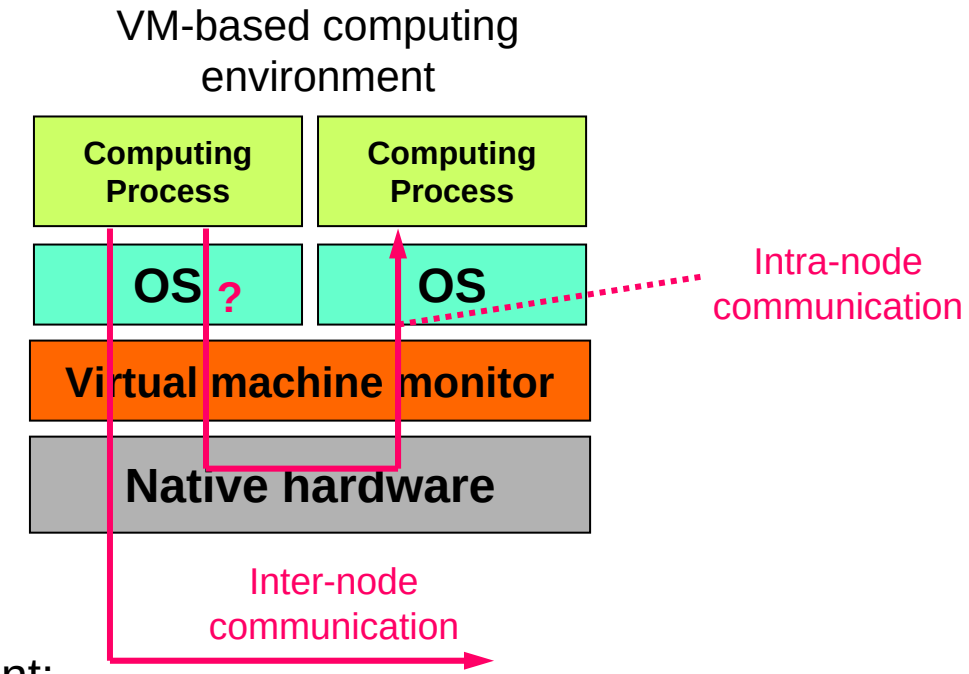
- Despite many promising features, VMs have not yet been widely used for HPC
- One of the most important reasons: perceived overhead from the virtualization layer
- Is this true?
  - CPU & memory virtualization:
    - Not really: HPC is full of non-privileged instructions, which can be executed natively
  - Communication I/O virtualization:
    - VMM-bypass I/O for network communication
    - Is that all?

A closer look at communication I/O



- Native environment running MPI job:
  - Inter-node communication through high speed interconnect
  - Intra-node communication through shared memory
    - More efficient: no network contention
    - Supported by MVAPICH/MVAPICH2, OMPI, etc …

A closer look at communication I/O



- VM-based environment:
  - Computing processes are hosted on **separate** VMs for scheduling flexibility
  - Inter-node communication through high speed interconnect
    - Support from VMM-bypass I/O – native level performance
  - Intra-node communication has to go through loop back as well
    - **Extremely undesirable especially with the wide-spread adoption of multi-core architecture!**

* Jiuxing Liu, Wei Huang, Bulent Abali, and Dhabaleswar Panda. High Performance VMM-bypass I/O in Virtual Machines. In USENIX'06

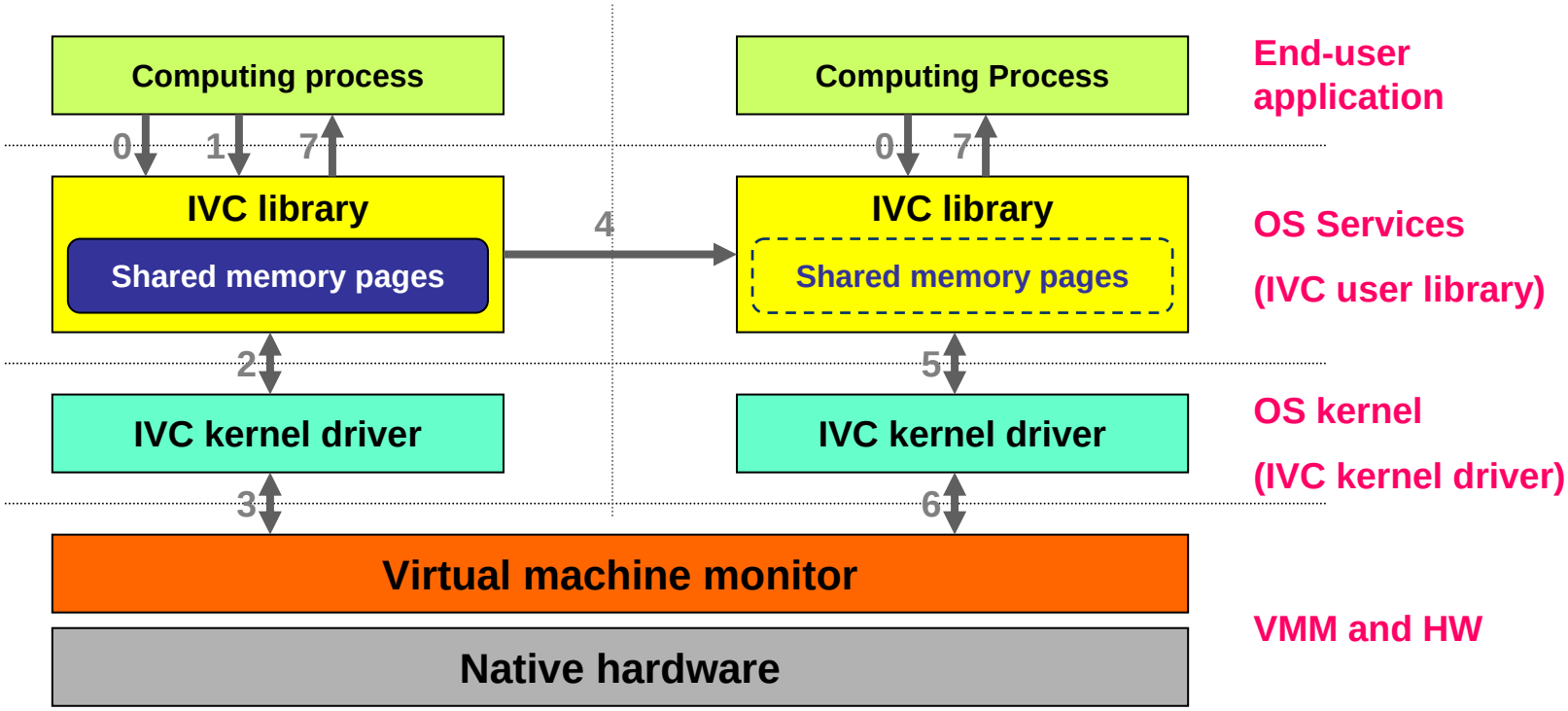
Our contributions

- Design IVC, an Inter-VM Communication library providing efficient intra-physical node communication through shared memory
- Hide all design complexities by designing MVAPICH2-ivc, a VM-aware MPI library
- Evaluate our design on multi-core computing systems, showing great potential for VM-based HPC

Inter-VM Communication

- Objectives
  - Providing efficient inter-VM (intra-physical node) communication through shared memory
    - How to setup shared memory region?
    - How to find peers on the same node?
  - Handling VM migration
    - How to tear-down/establish inter-VM communication?

Shared memory setup: a client-server model

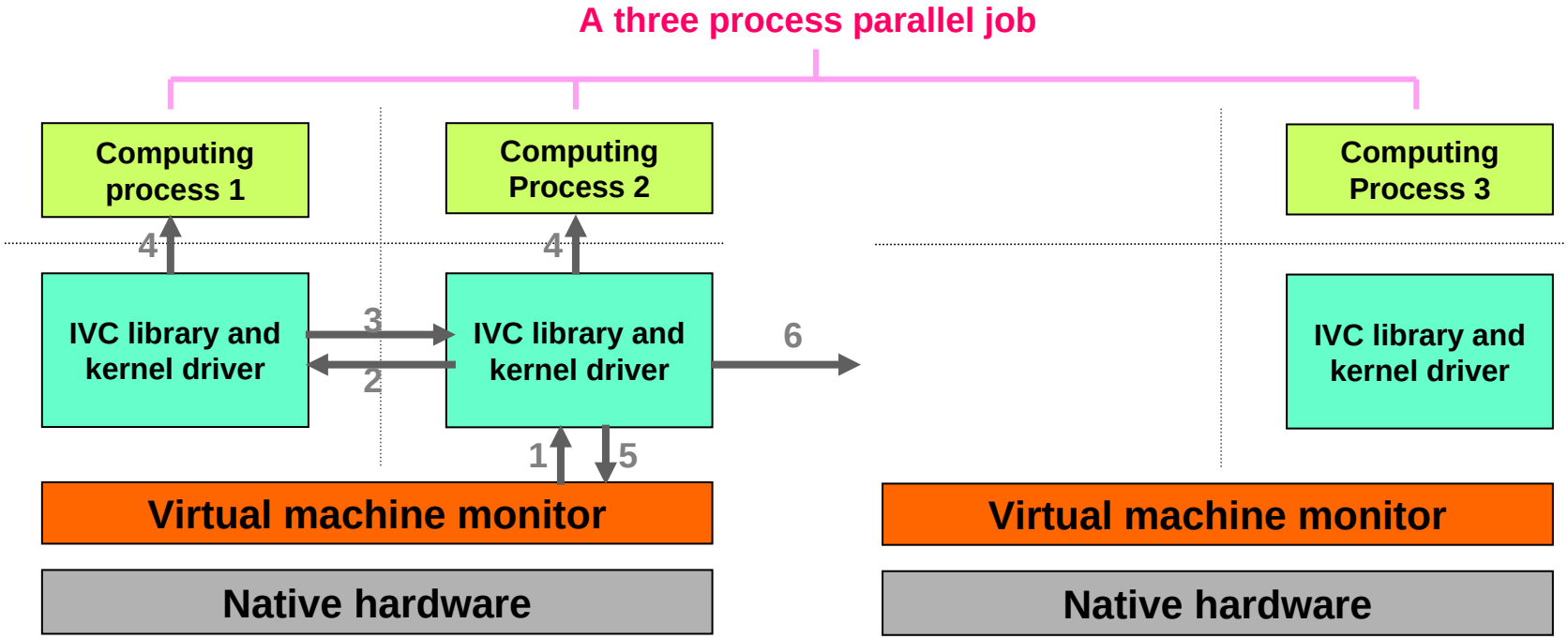


- Step 0: register – kernel drivers helps computing processes to find out peers on the same computing node
- Step 1: A user process initiates the setup process
  - Call into the IVC user communication library
- Step 2 & 3: IVC user library allocates shared memory space and grant page access to the remote VM through VMM
- Step 4: reference information is sent to the remote IVC library
- Step 5 & 6: Map the shared memory pages to process' address space
- Step 7: computing processes get notified

When VM migrates …

- IVC is a intra-node (physical node) communication library:
  - IVC connections to VMs on the original host must be torn down
  - IVC connections can be established to VMs on the new host
- Require peer coordination

When VM migrates …



- Step 1: IVC kernel driver on the migrating VM receives a callback when VM is about to migrate
- Step 2 & 3: all peers stop send operations and acknowledge
- Step 4: computing processes get notified
- Step 5: return from callback
- Step 6: migrate to the new host and establish new IVC connection

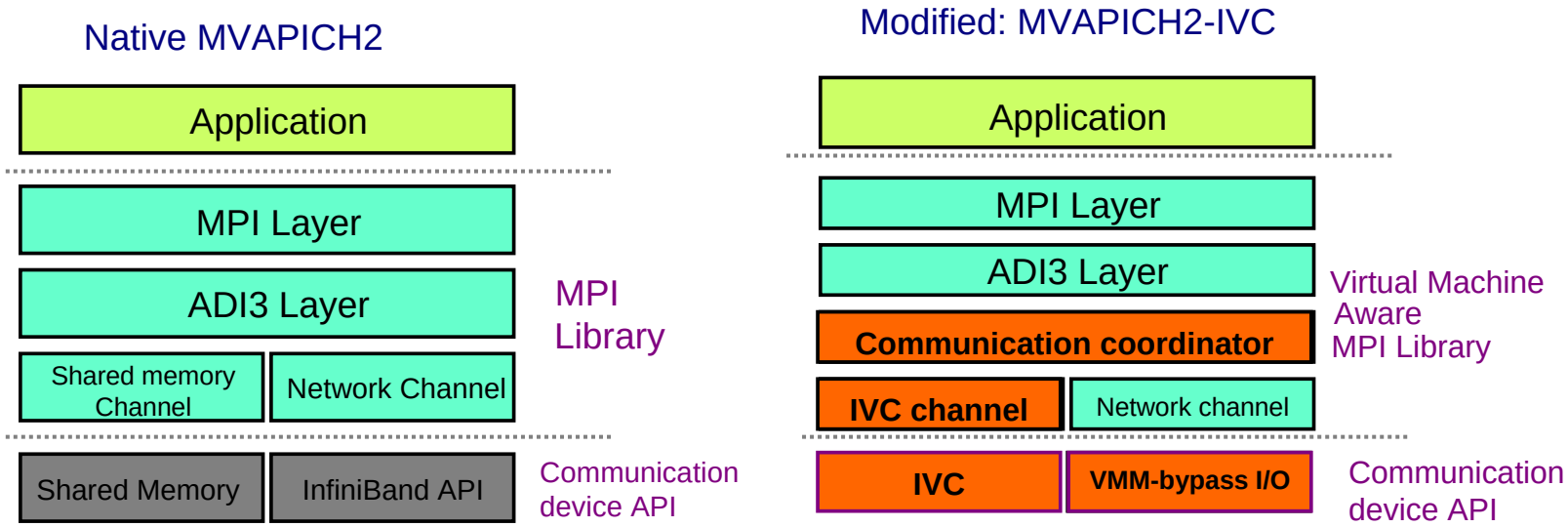
Now we have IVC ...

- Benefits:
  - Application can have efficient communication over shared memory, even when the computing processes are not in the same guest VM
  - Possible to support VM migration
- Concerns: application needs to
  - Written with our API
  - Take care of both intra- and inter-node communication
- Not a big deal!
  - Most applications are written in standardized APIs, like MPI
  - We can integrate our design into those API implementations

MVAPICH2-ivc: hiding the complexities

- Choosing MPI: the *de facto* standard for parallel programming
- MVAPICH2: a popular MPI-2 library over InfiniBand from our lab, used by 580 organizations world wide
- MVAPICH2-ivc: extends MVAPICH2, automatically choosing between IVC or network (IB) communication
- Hiding the complexities of IVC-specific APIs – transparently benefits user applications

Architectural overview



MVAPICH2 (native)

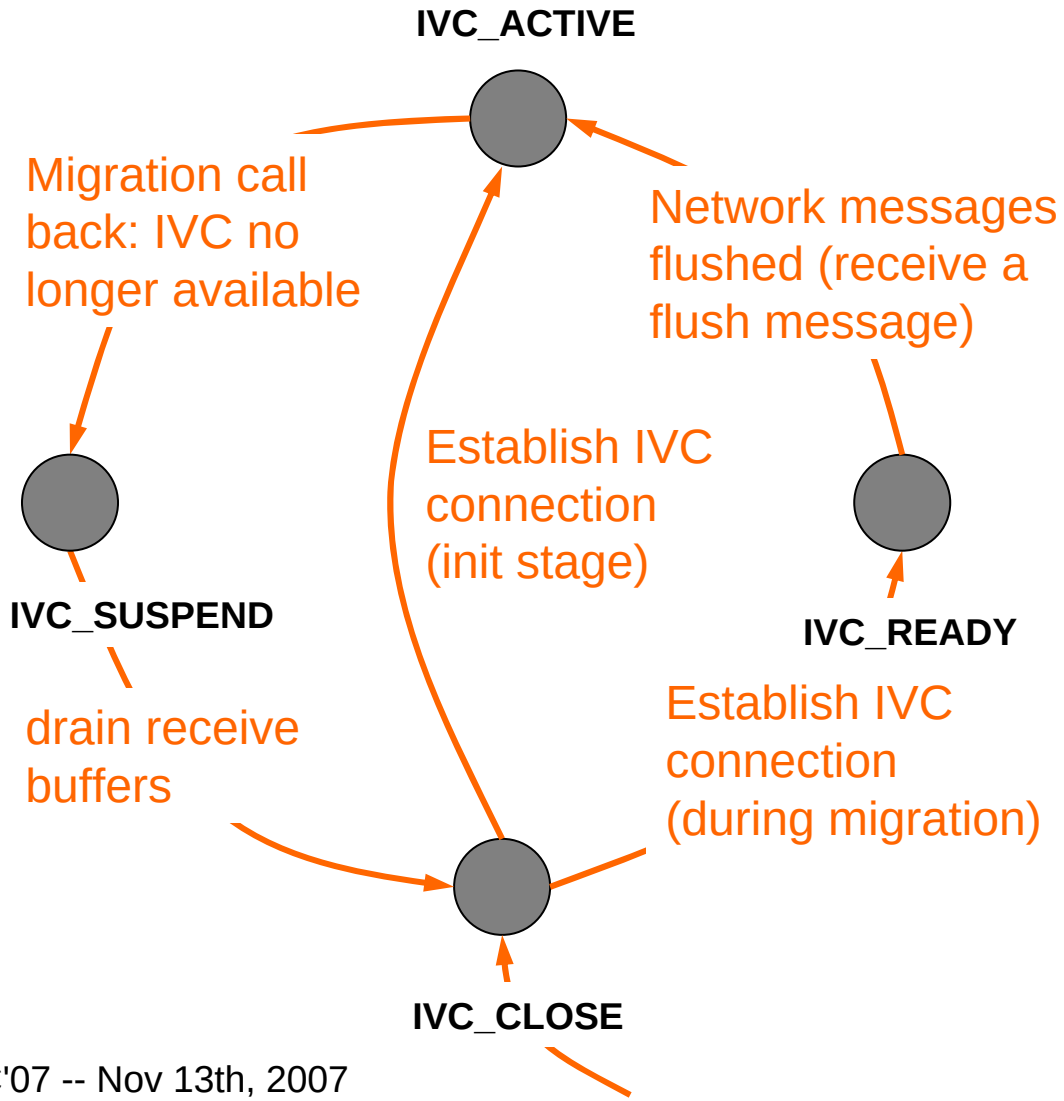
- ADI3 manages message delivery
- Shared memory and ADI3 channels are statically setup
- Shared memory channel communications over shared memory (OS provides mapping service)
- Network channel communicates over InfiniBand

MVAPICH2-ivc

- ADI3 manages message delivery
- Communication coordinator manages IVC and network channels setup (dynamic)
- IVC channel communicates of shared memory (IVC library/driver provide mapping)
- Network channel communicates over VMM-bypass over InfiniBand (transparent)

Handling VM migration

- **Key issue**: ensuring message in-order delivery when setting up and tearing down IVC connections during migration
- VC: virtual connections encapsulating communication mechanisms:
  - Network
  - IVC
- VC has four states:
  - ***IVC_CLOSE***: all VC start with this state
  - ***IVC_ACTIVE***: IVC connection is ready to use
  - ***IVC_SUSPEND***: IVC connection is being torn down
  - ***IVC_READY***: IVC connection is setup, but not ready to use due to in-flight message over network



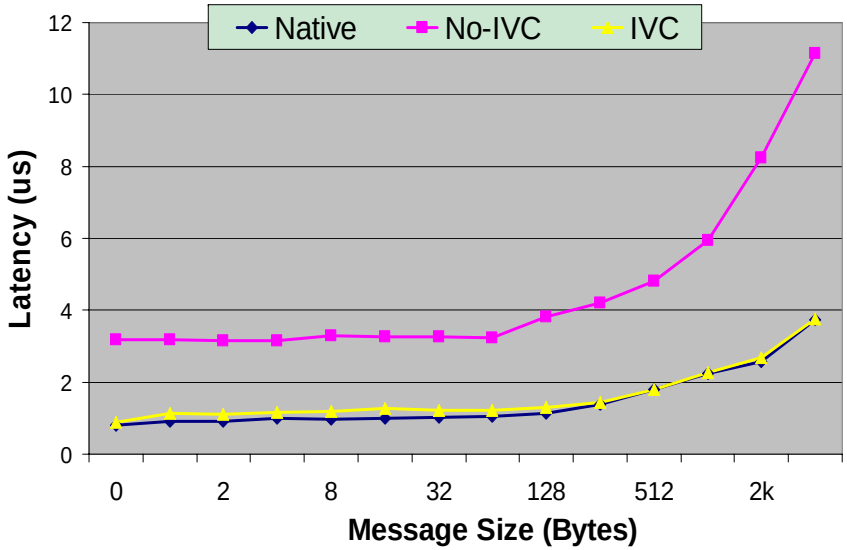
Now we have a MPI …

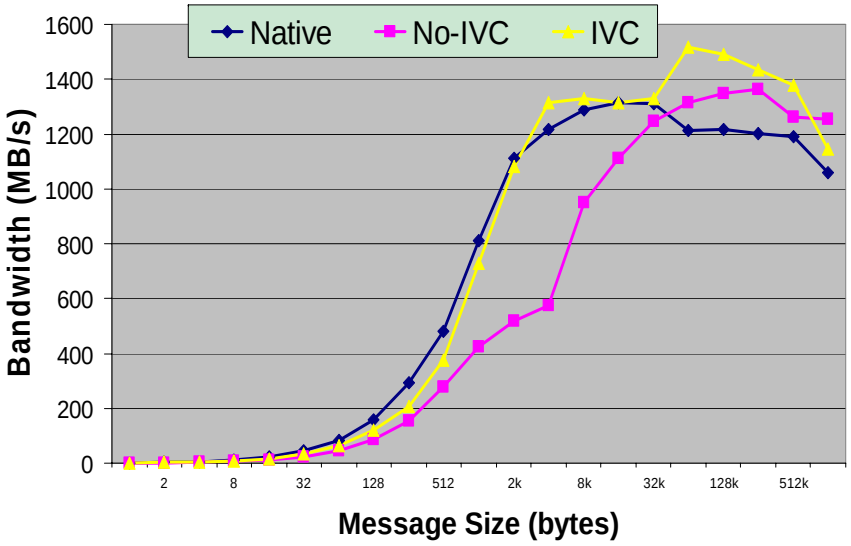
- Unmodified MPI applications can benefit from our design
- Regarding the performance concerns:
  - What's the benefit of IVC?
  - How does a VM-based environment with IVC compare with a native environment?

Experimental setup

- Testbed A: dual socket Intel Clovertown (Quad-core) processors, 4 GB memory, PCI-Express InfiniBand HCA
- Testbed B: 64 node dual socket single core cluster (32 Xeon and 32 Opteron), 2GB memory, PCI-Express InfiniBand HCA
- Xen-3.0, dom0 running RHEL 4
- DomU using ttylinux (tiny linux distribution)
- Configurations:
  - IVC: mvapich2-ivc running in VM-based environment
  - No-IVC: unmodified mvapich2 in VM-based environment
  - Native: unmodified mvapich2 in native Linux environment

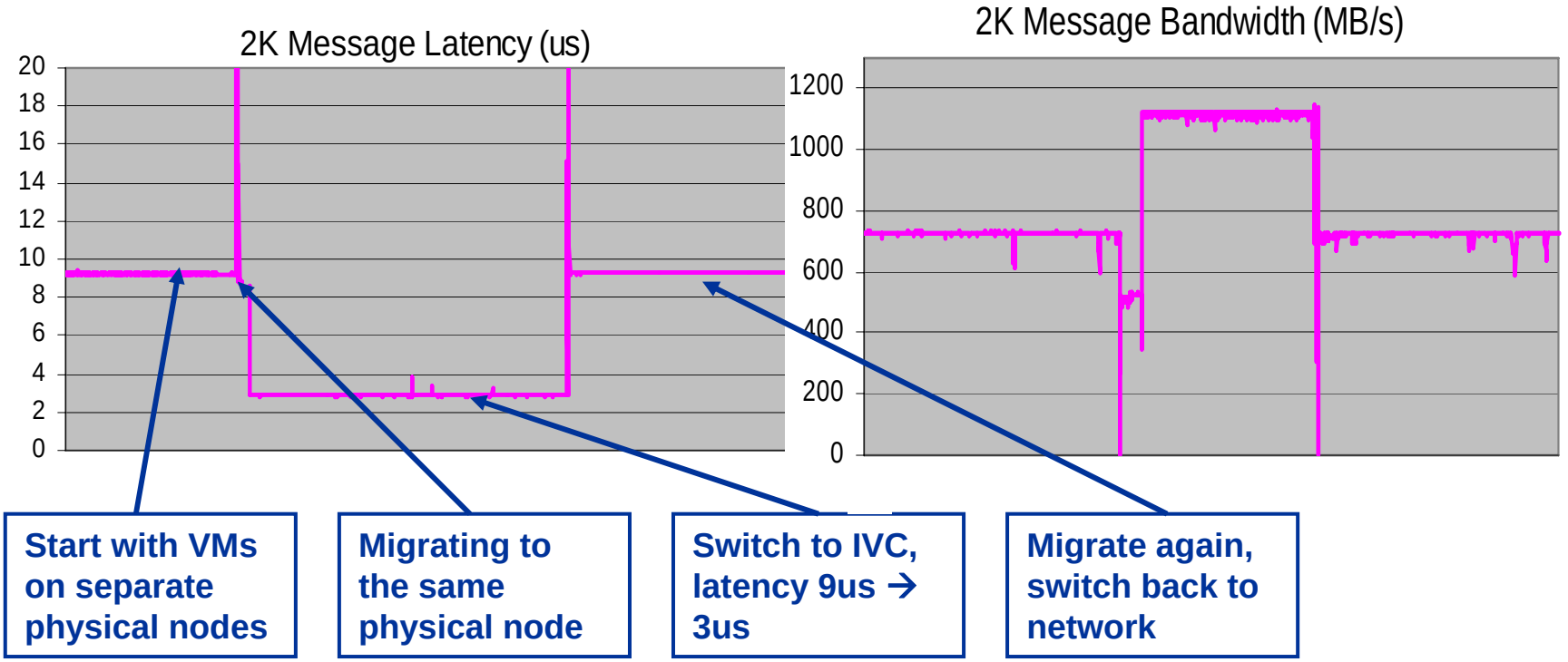
Latency and bandwidth





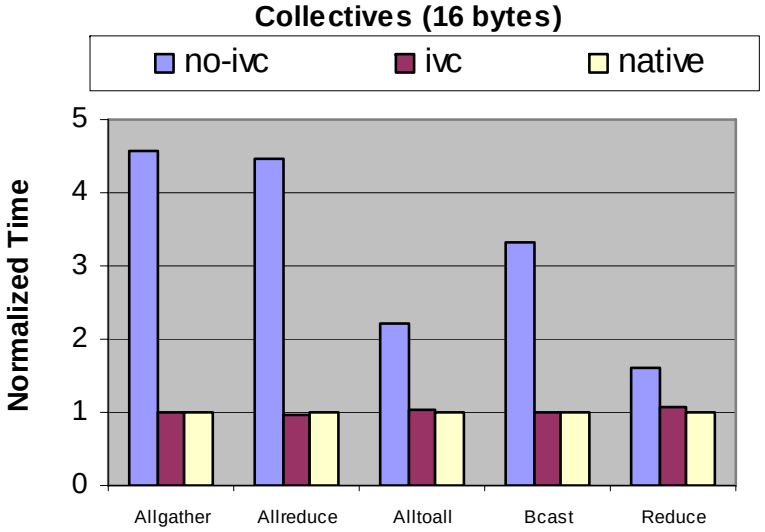
	Native	No-IVC	IVC
Latency	Sub-1us through Shared memory	~3.2us through IB loopback	Very close to native
Bandwidth	Much higher for mid-size messages	Getting better for large messages	Native-level performance

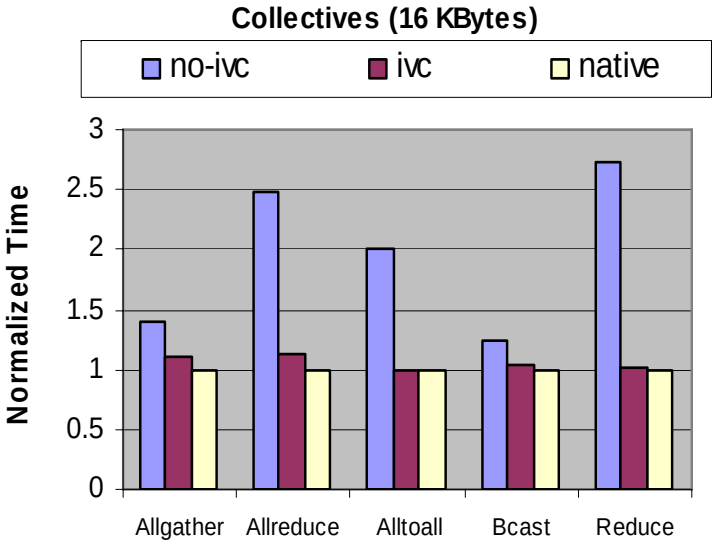
VM migration

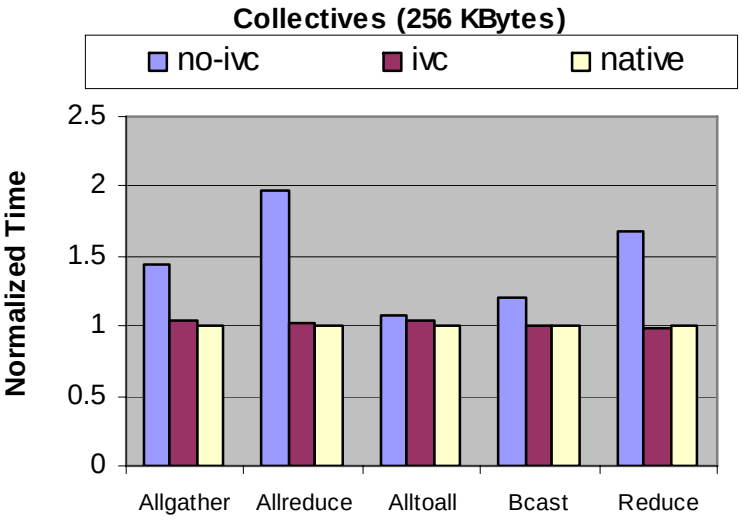


- MVAPICH2-ivc automatically switches to IVC whenever the target peers are on the same physical nodes
- Above two graphs show decreased latency and increased bandwidth when two processes in separate VMs are migrated to the same physical nodes

Collectives

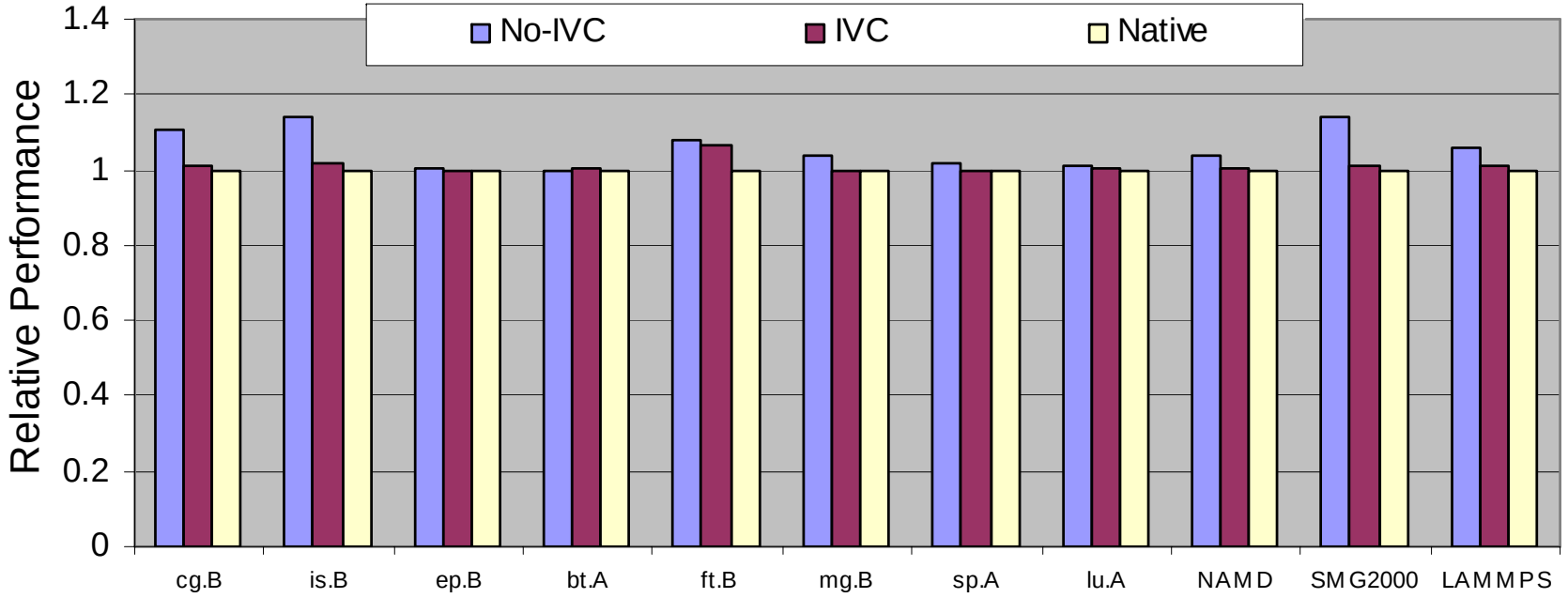






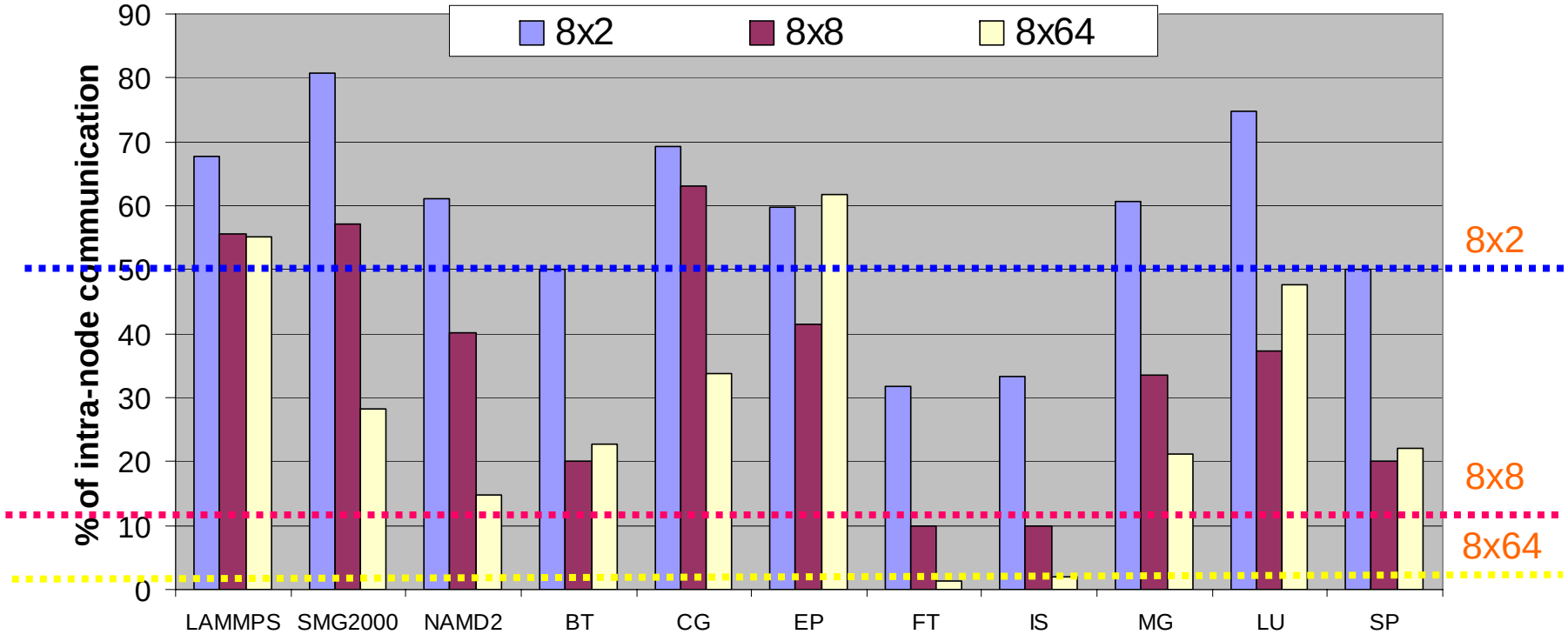
- With inter-VM communication, mvapich2-ivc largely closes the gap between native and VM based environments
- Results collected on 8-core systems using Intel MPI Benchmarks (IMB) (8x2)

Application-level benchmarks



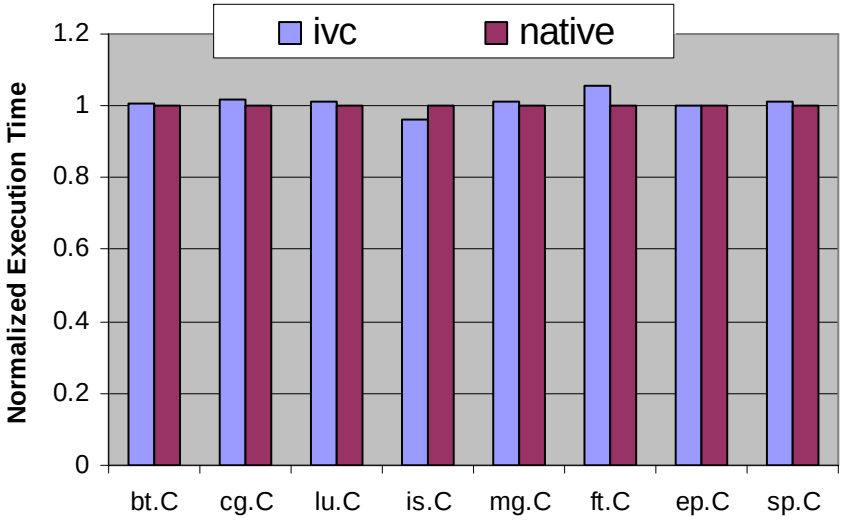
- Number taken on 16 processes
- Benefits of IVC show for several benchmarks, e.g. IS (11%), CG (9%), LAMMPS (5.9%), SMP2000 (11.8%), and NAMD (3.4%)

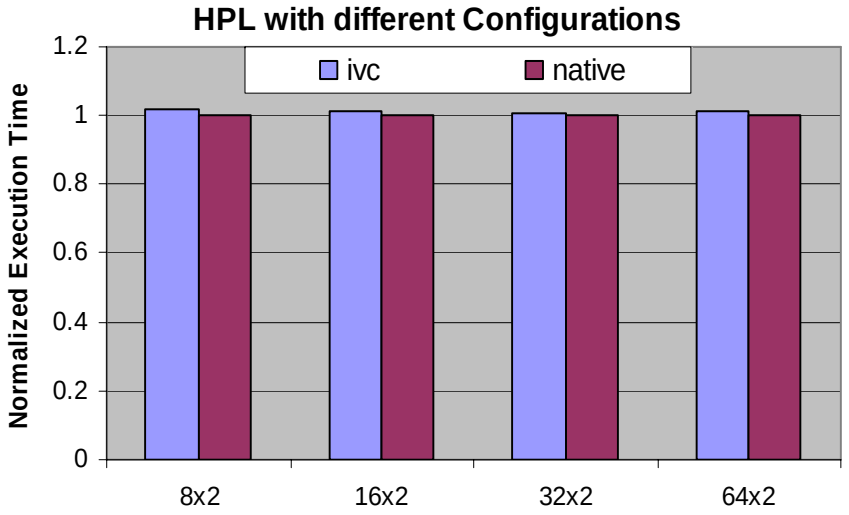
Larger scale?



- Based on individual benchmarks, intra-node communication is still an important part!
- Percentage of intra-node communication is well above average

Overheads on 64 node cluster





- Performance comparison on a 64 node dual processor cluster
- We do see very close performance (~1%)
- NAS-FT shows around 5% overhead with its large message all-to-all communication pattern

Conclusion

- We propose Inter-VM communication (IVC), allowing efficient shared memory communication between VMs
- We modify MVAPICH2 to hide all complexities and allow user applications to benefit transparently
- With our evaluation, we show: virtualization is NOT introducing much overhead
- With its benefits for system management, VMs are an attractive solution for HPC!

Future work

- More optimizations can be made to improve the performance of Inter-VM communication
  - Dynamically map user buffers to achieve one-copy communication
- Looking more into management frameworks for VM-based computing environments (load balancing, fault-tolerance …)

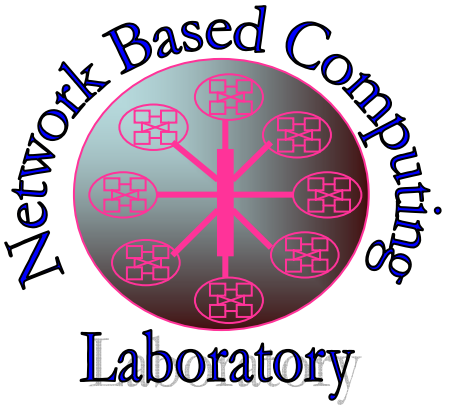
Acknowledgements

Our research at the Ohio State University is supported by the following organizations:

- Current Funding support by

- Current Equipment support by


Thank you!



Network-Based Computing Laboratory

http://nowlab.cse.ohio-state.edu/