Venice: A Software-defined Architecture for Data Center Servers

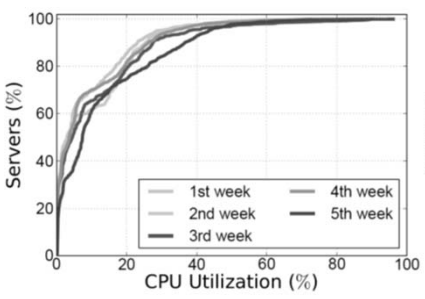
Rui Hou, Jianbo Dong, Boyan Zhao, Michael C. Huang, Sally A. Mckee, Lixin Zhang

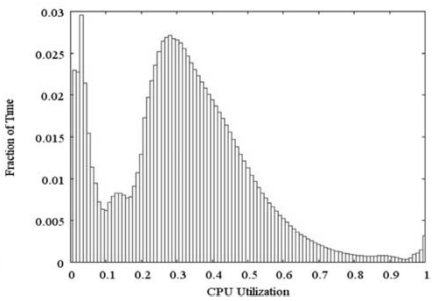
Institute of Computing Technology, Chinese Academy of Sciences

中国科学院计算技术研究所
Institute of Computing Technology,Chinese Academy of Sciences

Background

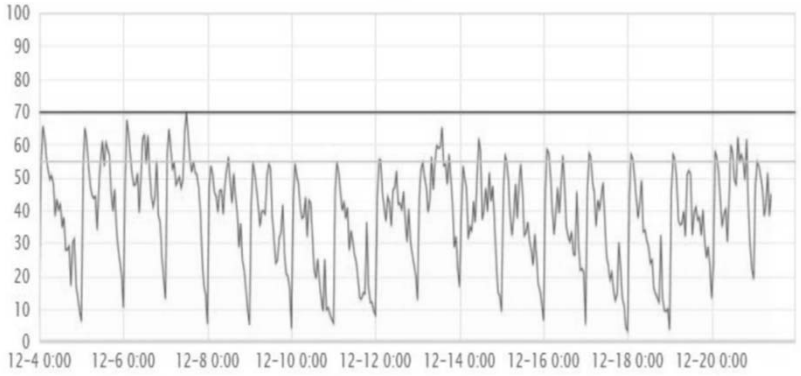
- **Resources utilizations of Data centers are very low [twitter, and google]**
  - **Isolation of latency-sensitive applications**





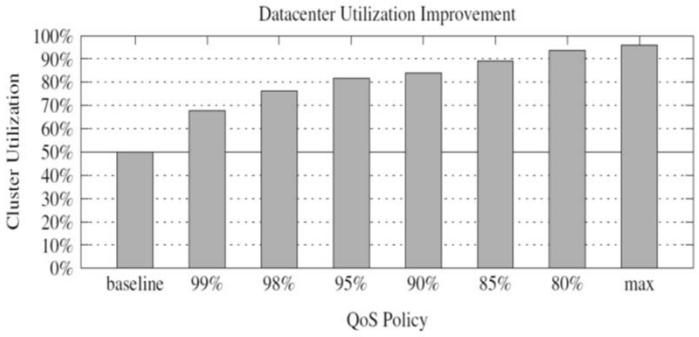
Background (cont.)

- Resources utilizations of applications present timing-varieties [Taobao]



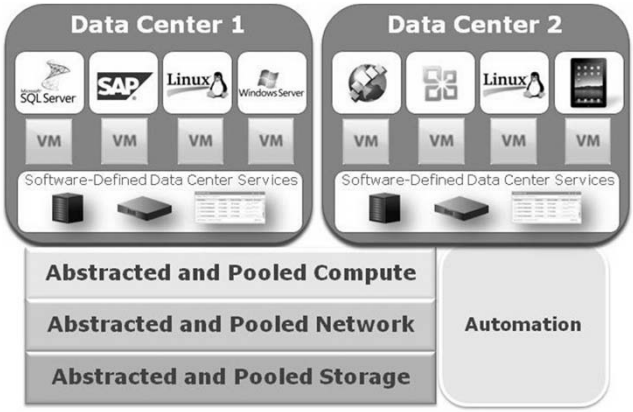
Background (cont.)

- To tackle the problem, task co-location techniques are proposed [J. Mars]



Background (cont.)

- And, of course, virtualization techniques: Software Defined Data Center [VMware]

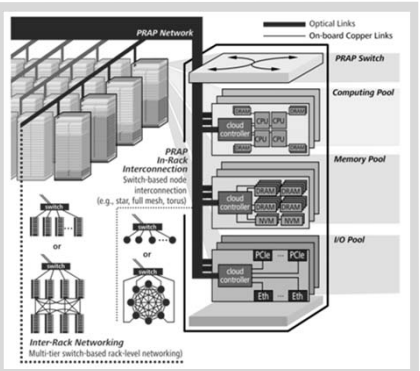


Motivation

- Root cause of lower utilization:
  - Emerging applications call for flexible and high-efficient resources provision
  - Architecture of commodity consolidated data centers has remained static
  - Mismatches between conventional cloud infrastructure architectures and their workloads.

The proposal of Venice Architecture

- A family of cloud server architectures
  - Providing a set of resource-joining mechanisms
- Example implementations
  - Centralized resources joining
  - Distributed resources joining

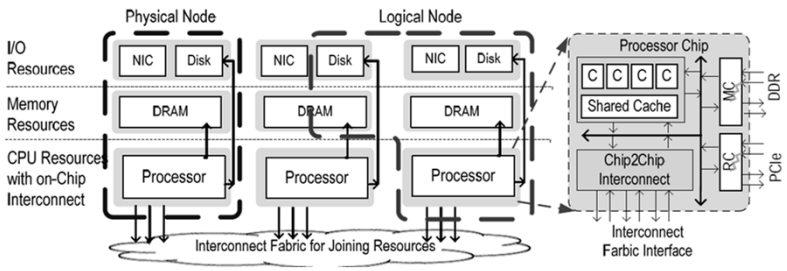


Centralized resources joining:
Resources-lack node should be able to access the remote resources pools efficiently and transparently.

Collaboration with Huawei, "High Throughput Computing Data Center Architecture Thinking of Data Center 3.0"

Institute of Computing Technology, Chinese Academy of Sciences

The proposal of Venice Architecture



- The distributed resources joining
  - Here, resources-lack node should be able to utilize idle remote resources *easily* and *efficiently*
    - *transparently to user-level applications*
    - *with tolerable performance degradation*

8

Institute of Computing Technology, Chinese Academy of Sciences

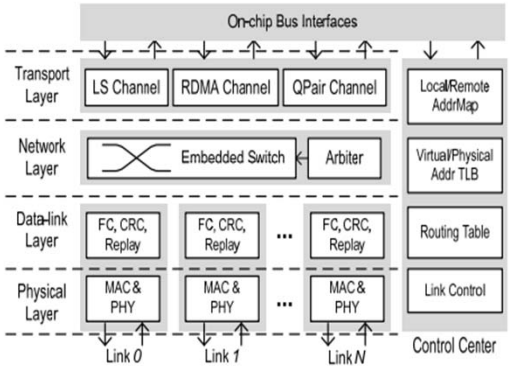
Three KEY layers for resource sharing

- Resource-sharing fabric
  - □ low latency and high bandwidth
- Mapping and joining mechanisms
  - □ user-application transparency
- Management runtime
  - □ intelligent resources management

Fabric design decisions

- On-chip fabrics
  - To minimize latencies
- Multi-modality
  - To adapt to various data transmission patterns, including explicit communication and remote memory accesses at both coarse and fine granularities
- Transparency
  - To provide productive interfaces that not only minimize user-level runtime overheads but require few changes to user programs

Hardware protocol



Control center:

address translation, switch control, and link control

Data-link & physical layer :

point-to-point reliable transmission

Network layer:

embedded switch for packet forwarding

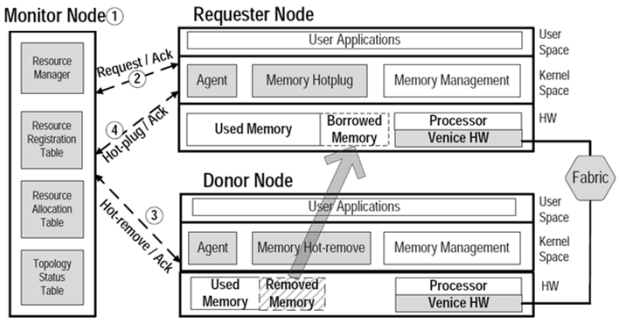
Transport layer :

Load/Store channel, resources sharing support in hardware level

RDMA channel, resources sharing support in kernel level

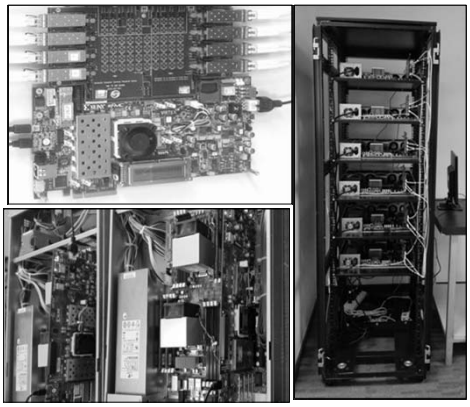
Qpair channel, resources sharing support in application level

Management runtime



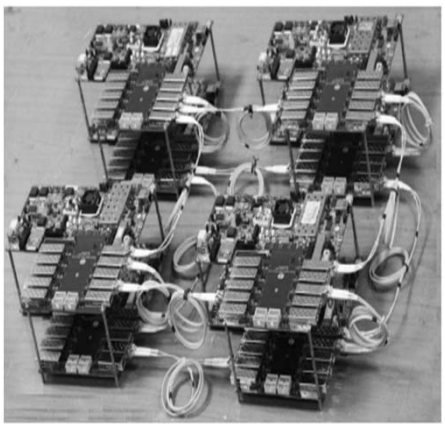
- A running example
  - An agent on each node periodically collects resource usage information and reports it to a central MN.
  - When a node needs resources beyond its local capacity, it sends a request to the MN, which selects appropriate donor node(s) to provide resources.

Evaluation platforms – embedded + X86



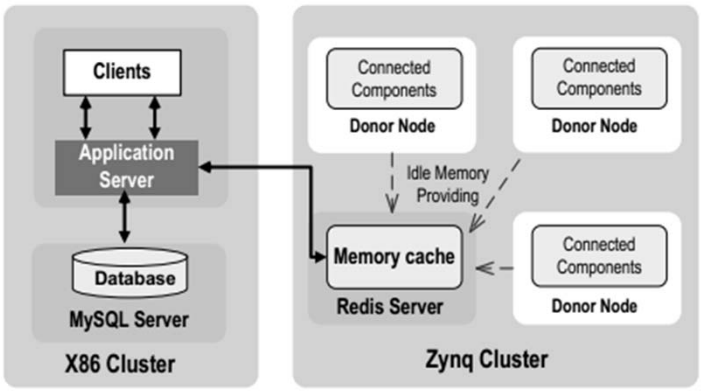
- VC709 board connect with X86 server via PCIe slot
- Venice + PCIe core on VC709 board
- X86 server accesses another server's memory by VC709 board

Evaluation platforms - embedded based

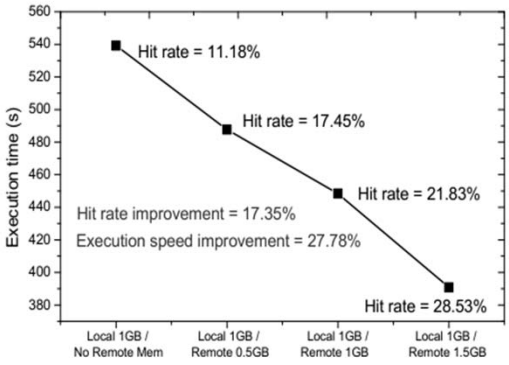


- 3D cube-connected Xilinx ZC706 boards with ARM Cortex-A9 processors
- Venice substrate in programmable logic connecting the ARMs via AXI buses with cache coherence support
- The system runs a complete software stack, including Linux (Linaro 13.09)

Configuration - mini data center



Evaluation results – mini data center



- **The Redis hit rate improves with increases in memory resources.The execution time has a 28% performance improvement.**
- **CC experiences no performance degradation because the memory traffic caused by remote sharing is insignificant**


Institute of Computing Technology, Chinese Academy of Sciences
Thanks
17