Software Acceleration in Hybrid Systems

Xiaoqiao (XQ) Meng
IBM T.J. Watson Research Center
May 4, 2011

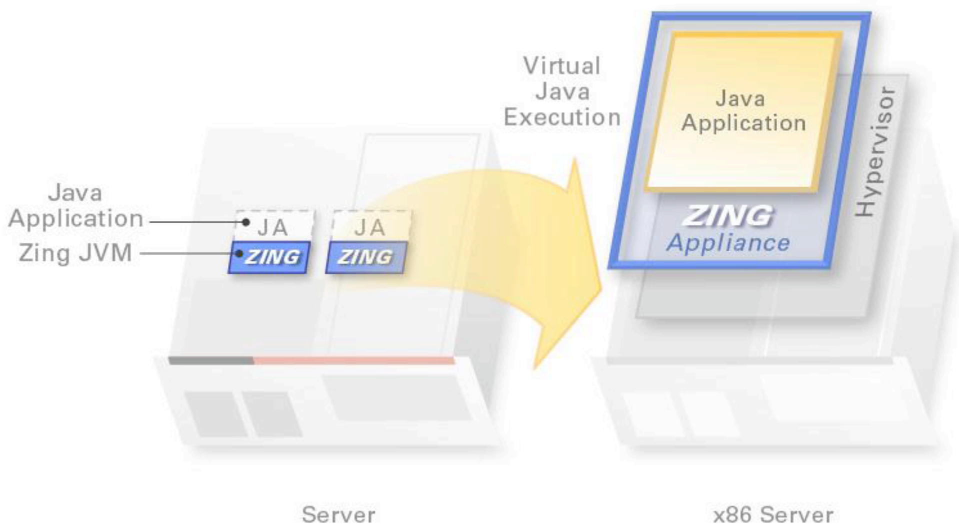
Hybrid Systems

- A distributed system consisting of heterogeneous computing architectures
    - E.g., X86, PowerPC, ARM, GPU, FPGA
- In a broader sense, a hybrid system also includes heterogeneous storage, I/O and communication devices
- Advantages
    - Complementary strengths and weakness
    - Cost-effective
        - Mixed low-end and high-end computing platforms to be elastic. E.g., the TianHe supercomputer uses Intel Xeon + AMD GPU hybrid system to achieve high floating point performance in economical manner
- Use cases
    - General-purposed computing systems
    - Fit-for-purpose appliance

Example: Zing platform from aZul

- Zing: Elastic Java runtime platform
- The Zing Java Platform consists of 4 main components: a 100% Java-compatible JVM which installs and launches like any other commercial JDK/JVM; a highly optimized runtime platform (i.e. kernel) packaged as an easy-to-install virtual application;an integrated,application-aware resource management and process controller and a true, zero-overhead, always-on

  production-time diagnostic and tuning tool integrated into the Zing JVM and appliance



Leveraging heterogeneity

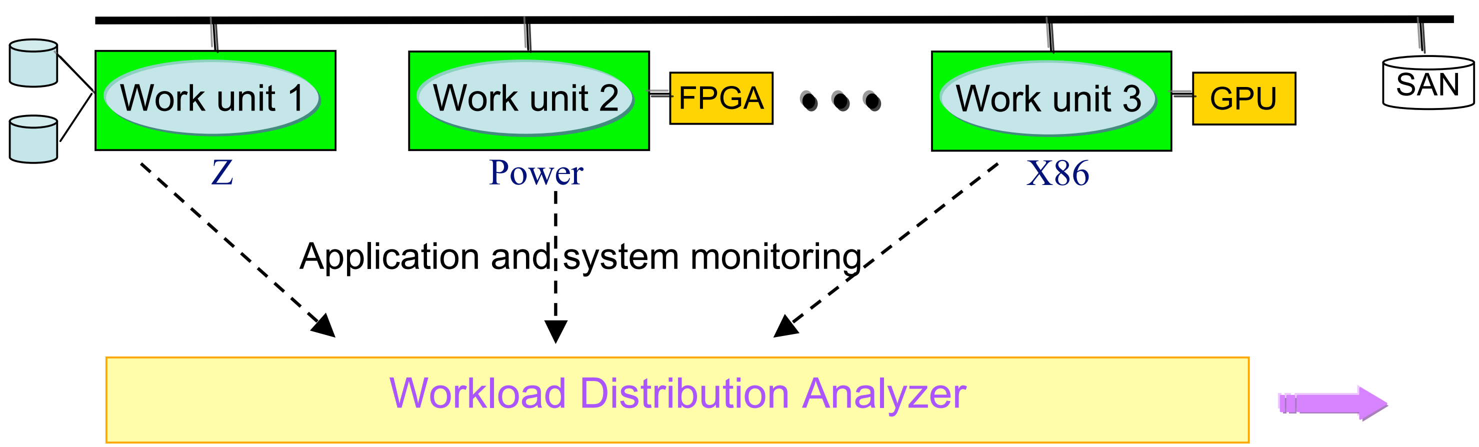
- Essence of software design in hybrid systems: it is possible to leverage the heterogeneity to accelerate software performance
  - Scenario 1: migrate computing-intensive work units from CPUs with expensive MIPS to CPUs with cheap MIPS
  - Scenario 2: Place multi-thread work unit on CPUs with good SMT supports and multi-core

Challenges

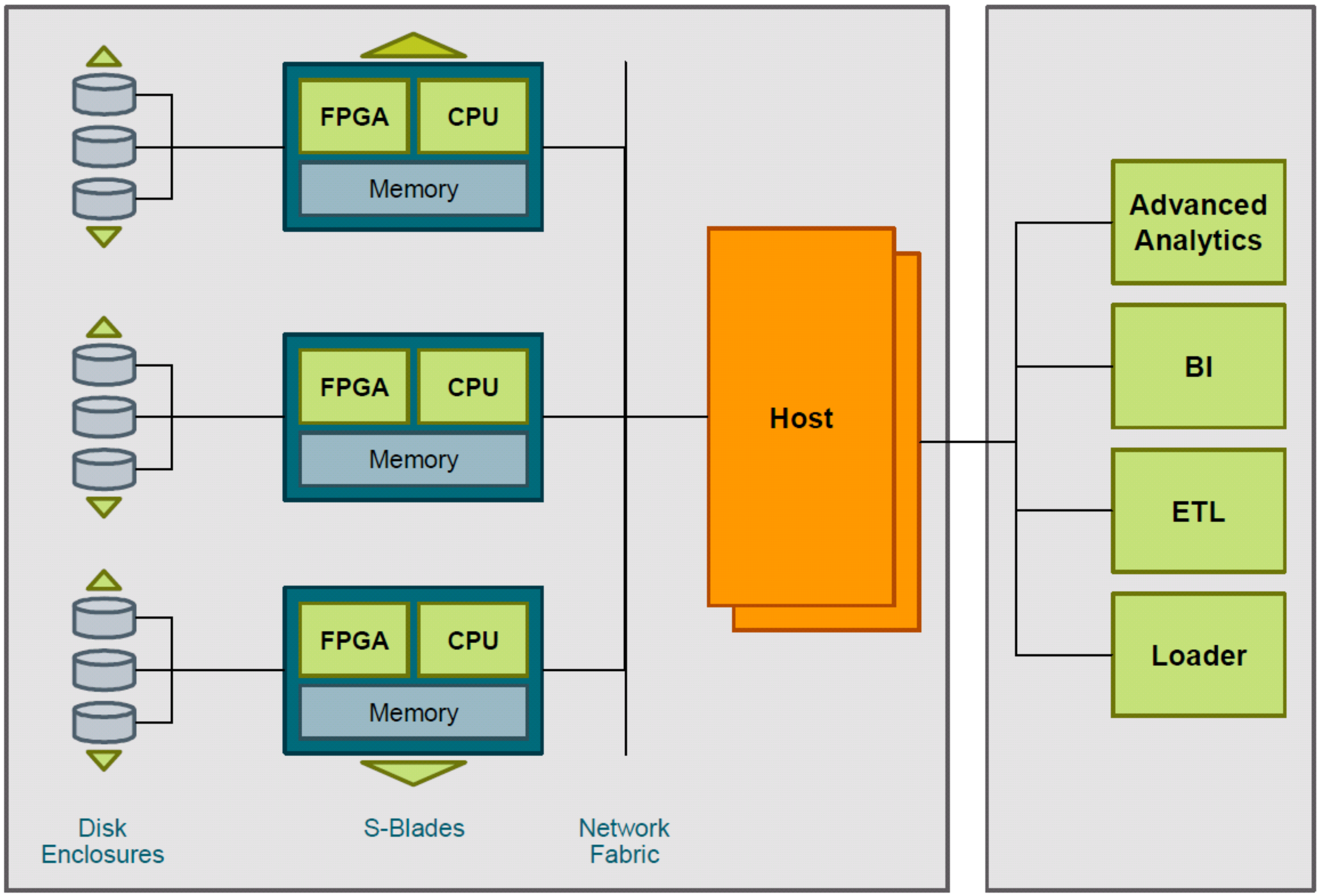
- How to detect acceleration potential?
- What software technology to realize the potential?

Affinity based Workload Distribution

- Goal: In hybrid systems, distribute workload in run-time by leveraging affinity
  - **Hybrid system**: A computing system that combines general and special purpose machines
  - **Affinity**: A work unit more efficiently processed on one platform than the other
  - Analytically determine workload distribution that optimizes performance
- Technical challenges
  - Identify appropriate application/system metrics to characterize workload
  - Workload characteristics change over time
  - Recognize the affinity of workload to platforms during runtime
  - Conflict between affinity and load balancing



Example from Netezza



Workload Distribution Analyzer

- Detect re-distributable work unit
  - For Java workload, a localizable JNI call is a re-distributable work unit
- Use domain knowledge and machine learning techniques to determine affinity of work units
  - Domain knowledge
    - E.g., in general Power has better Decimal Floating-point support than Z. So a Math module has better affinity on Power than on Z
  - Machine learning techniques
    - *Affinity clustering*: work units referencing similar data are clustered together to be executed on the same hardware
    - *Supervised learning*: by learning from work units affinity to Power, infer whether other work units have affinity to Power
  - Una-May's ML approach might be a very good fit
- Determine workload re-distributing strategy to optimize performance
  - Principle of data locality
    - E.g., if a work unit has much input data on Z, it should remain on Z instead of migrating to Power
  - Tradeoff between locality and overall load balancing
    - Respect data locality while maintaining load balancing across hybrid systems

Existing simple practice for affinity based workload distribution

- Workloads that benefit from being on z
  - Workload that access significant amount of z data
    - If they are running on Power or x86, they can be moved to z
    - Z data can be provided to apps in a much more efficient manner
  - Workloads that require strong single threaded performance (z has the fastest threads in the industry)
  - Workloads that require strong security and unique quality of service
- Workloads that benefit from being on Power7/massive multi-threaded platforms
  - Long running applications with little or minimal data requirements
  - Multiple parallel threads that benefit from SMT on Power
  - Threads that require significant amount of main memory
  - Threads that require memory bandwidth
  - Analytics algorithms that are that are computation intensive (e.g., floating point operations)

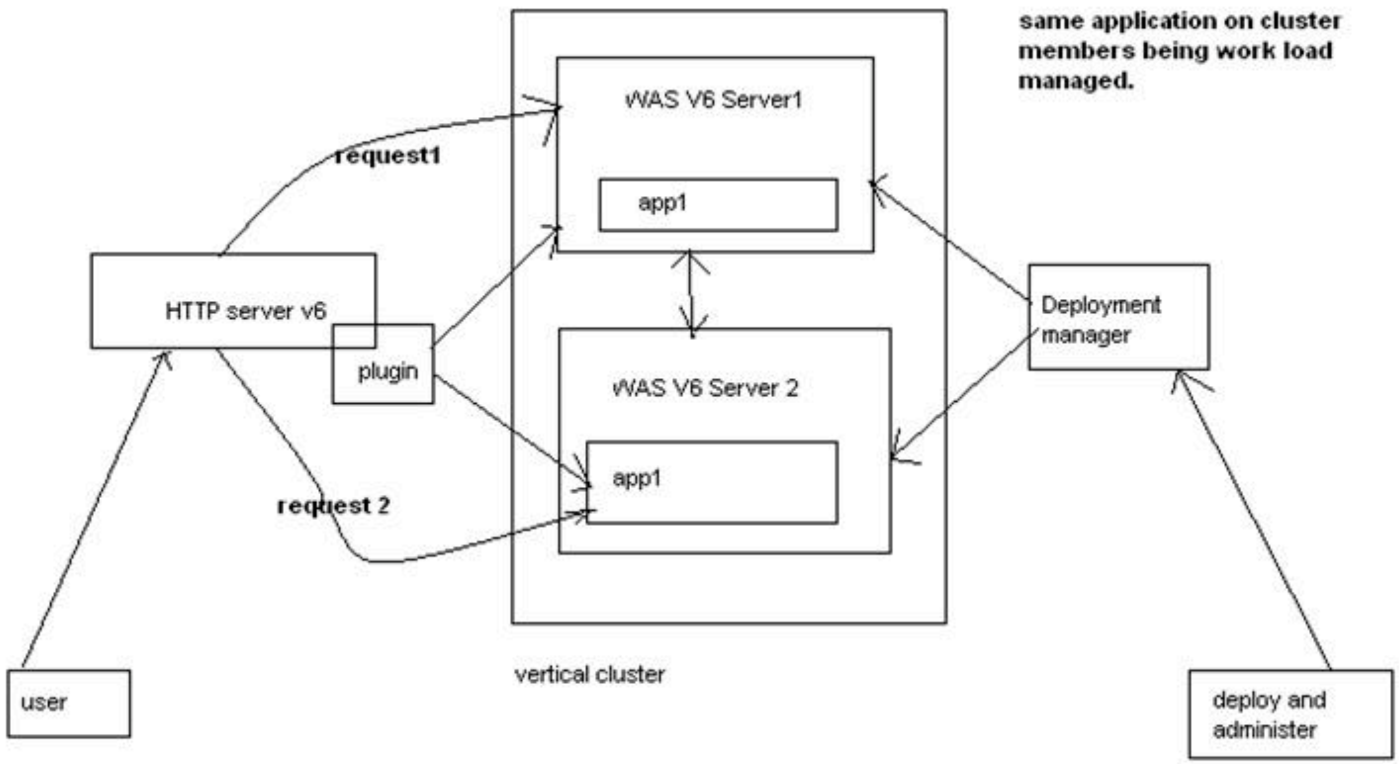
Building a hybrid system with diversified benchmarks

- Design and build a hybrid system testbed
  - Include major computing architectures
- Extend from computation-intensive, HPC types of benchmark to a wider range of benchmarks, including:
  - Web server
  - Java client/server (various Java middleware)
  - Mail server
  - Network file systems
  - Etc.
- Initially, we may focus on multi-tier benchmark and study performance acceleration within one-tier

Technical challenges

- Automatically inferring dependency graph
  - Derive input/output data flow
  - Infer dependency among work units
- Online learning efficiency for an independent work units
  - Quantify affinity/stickness. Consider security as extra constraints
- Combining dependency graph and efficiency learning to infer acceleration ratio
- Lightweight, cross-platform measurement tools
- Work with legacy applications without much code change

Websphere WLM



ARCHITECTURE OF VERTICAL CLUSTER