HIGH PERFORMANCE PROCESSORS

for Communication and Media

Harrison Zou

FreeBSD Developer Summit

May, 2009

Agenda

- Introduction of RMI Corporation
- Briefing on Multicore Multithreading Processor Architecture
- Software Considerations and FreeBSD Port

XLx Applications – *End-to-end Solutions*

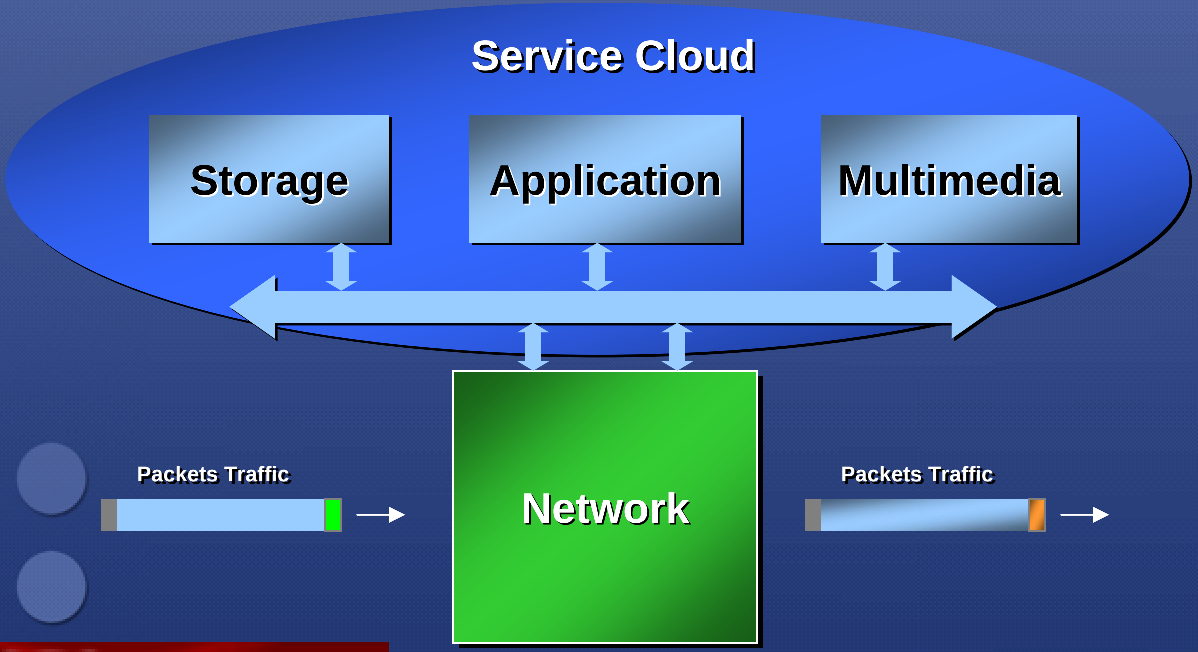
	XLx Configuration	Application Examples	Throughput
Infrastructure	4 – 32 Cores	*WiMax ASN Gateway* *Service Provider Routers* *Security Appliances* *Server Load Balancing* *LTE Channel Card* *GGSN and SGSN* *Base Station Controller (BSC)*	*10Gbps – 160Gbps*
Enterprise	2 – 8 Cores	*Enterprise Switches and Routers* *Security Appliance* *Storage Appliances* *Cloud Computing* *Data Center Ethernet* *Server Acceleration* *WLAN Switches and APs*	*2Gbps-40Gbps*
SMB and Home Networks	1 – 2 Cores	*Broadband Access Gateway* *Secure Wireless Router* *Network Attached Storage* *Security Appliances* *Network Video Recorder* *General Purpose Control Processing*	*100Mbps-4Gbps*

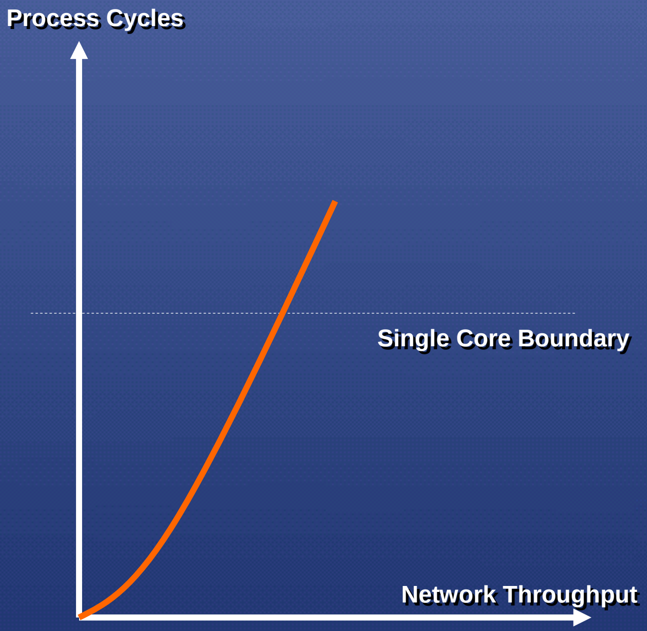
FreeBSD

RMI

Motivation of Multicore Network Processor

- Increasing Network throughput
- Increasing Processing cycle per packet
- Increasing Performance per watt per dollar
- Emerging Network Applications
- C Programmable





The Challenges of Multi-core

- **Multi-core design has become mainstream**
  - **Tremendous success… many scalable, high performance systems are shipping in production today…**
  - **… but, not all multi-cores are created equal**

- **Primary development issues**
  - **Software migration from single- to multi-core**
  - **Meeting the market demand of high performance**
    - Translating CPU horsepower into delivered performance
    - Inevitable consequences of large memory footprints
    - Unexpected demand for CPU resources grinds system to a halt
  - **Achieving system level scaling**
    - Devise unified hardware and software architecture for an entire family of products
    - The problem of maintaining multiple code bases

The Challenges of Delivered Performance

- **Multicore SOC's face a major challenge: device driver has become a major overhead**
  - **Driver overhead becomes overwhelming with large CPU core counts and high-speed peripherals**
    - Traditional peripherals do not scale with high CPU count due to synchronization and interrupt overhead
    - Even the most advanced CPU core will deliver poor system-level performance – due to bottlenecks to its peripherals
    - This problem requires a revolutionary on-chip interconnect
  - **RMI architecture features Fast Messaging Network to address this issue**
    - Conserves CPU cycles for real application, instead of driver overhead
- **Net result: maximum delivered performance using RMI architecture**

Key Architectural Differentiation

Current Product

Fine-Grained Multi-Threading MIPS64 Multi-Core

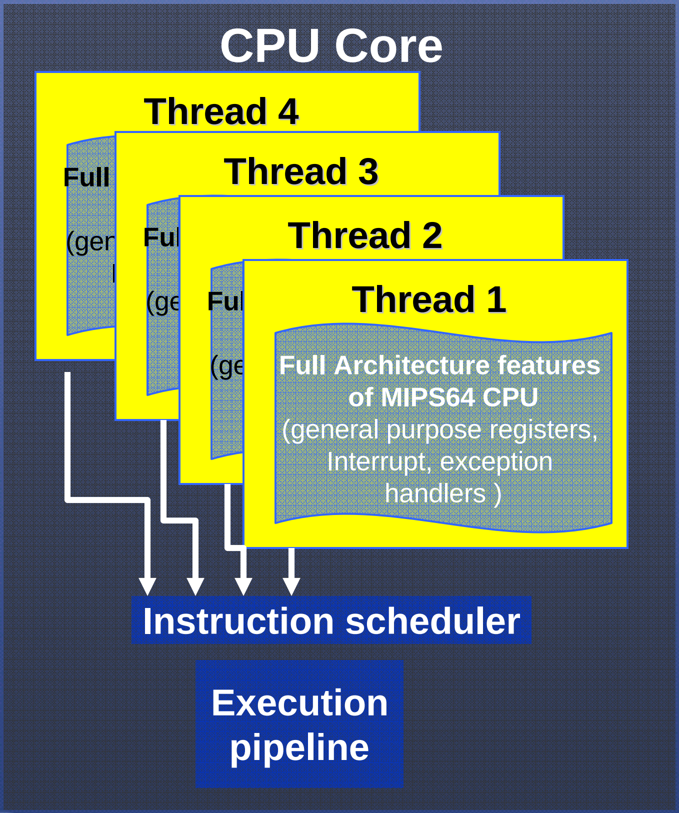
Fast Message Network

Superscalar OOO MT MIPS64 Multi-Core

Central Message Switch

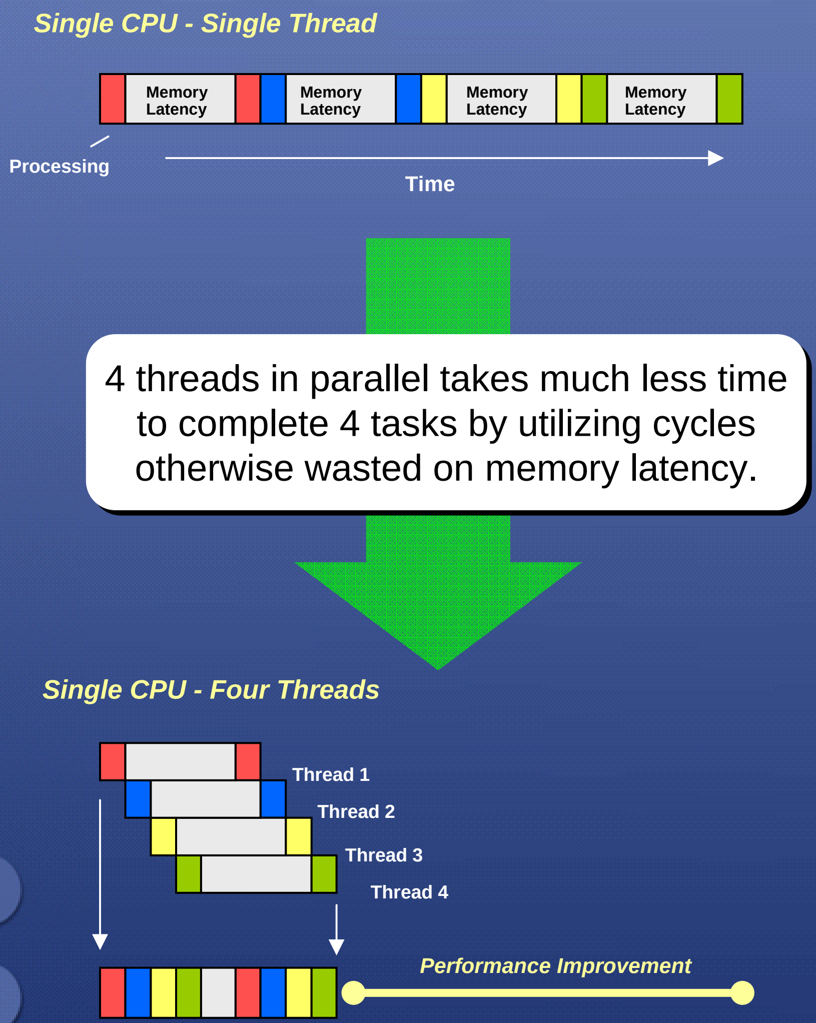
Next Generation

Fine Grained Multithreading


CPU Core
Thread 4
Thread 3
Thread 2
Thread 1
Full Architecture features of MIPS64 CPU
(general purpose registers, Interrupt, exception handlers)
Instruction scheduler
Execution pipeline

- Each CPU core contains four full-fledged CPU's
- We call them threads or vCPUs
  - Each one has its own unique register set
  - OS's and Applications see each one as an independent CPU
  - For example, Linux comes up as 32-way SMP in an 8-core XLR
- It's done in hardware
  - Switch from one vCPU to another every single cycle
  - The context switch is immediate without cycle loss

Benefits of Multithreading



4 threads in parallel takes much less time to complete 4 tasks by utilizing cycles otherwise wasted on memory latency.

- **Improves performance by hiding memory latency**
  - **When one vCPU (thread) stalls, the next one takes over**
  - **CPU usage is maximized**
- **Without threading, CPU's *will* stall out**
- **vCPU's (threads) consume much less area and power at a given performance level**

Memory Latency – Now a Primary Issue

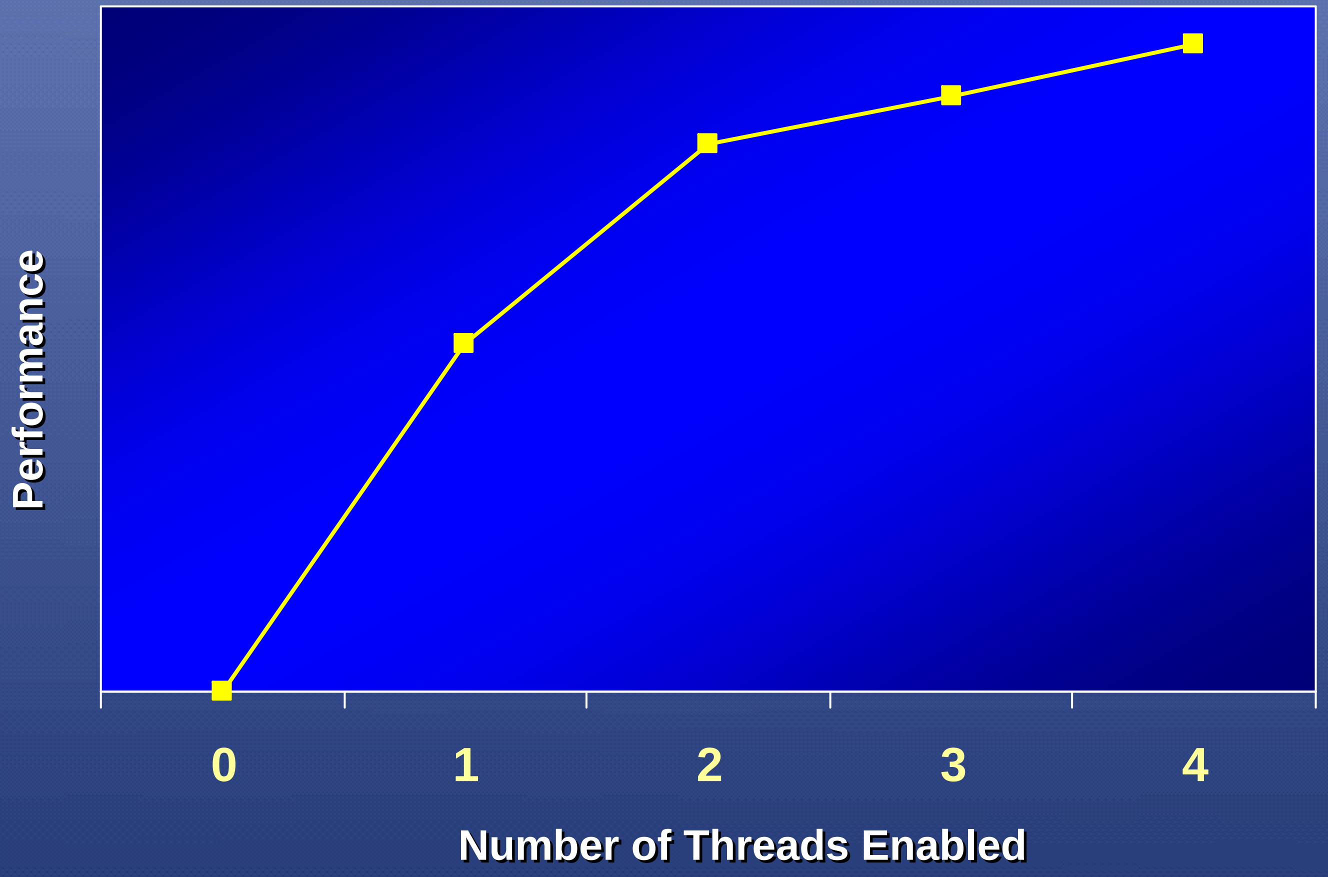
- Real-life networking and security applications typically require a large memory footprint
  - Memory latency becomes the killer
  - Highest frequency CPU does not solve this problem

- However, RMI architecture experiences little drop in performance
  - Multithreading effectively hides memory latency

Route Table Size	XLS 1 Core	Intel Xeon
Two Route Entry		
10 K Route Entry	Only 10% drop	66% drop

Real carrier Network Typically has >200K routes

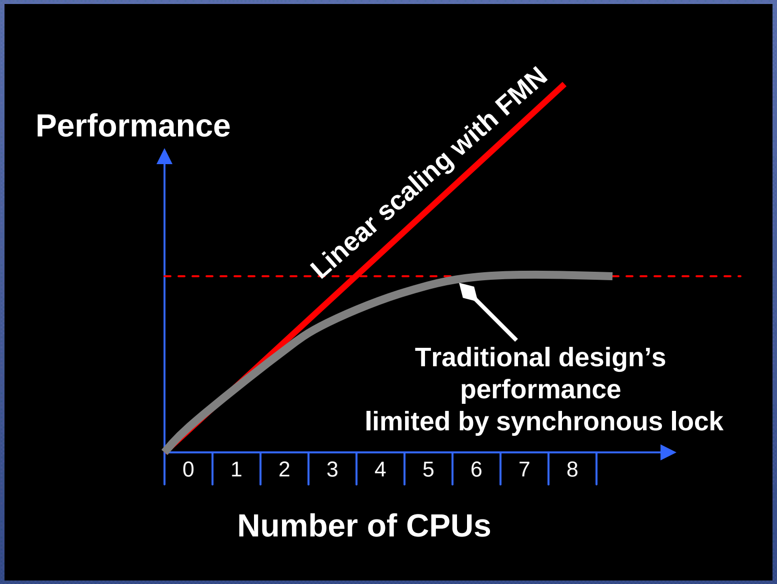
Thread Scaling within A Single Core



Fast Messaging Network Overview

- **FMN is an on-chip interconnect designed specifically for passing messages between CPUs and peripherals**
  - **Light-weight software driver**
    - RMI-specific instructions for sending/loading FMN messages
    - Few instructions to send (or receive) a message
  - **Prevents unnecessary context switches**
    - FMN provides interrupt-free method to receive message arrival notification
  - **Lockless**
    - Hardware buffers are reserved for each sender
    - As such, multiple senders can simultaneously transmit at each cycle. No need for software synchronization
  - **High bandwidth on core clock frequency**
  - **Very low latency (ns)**

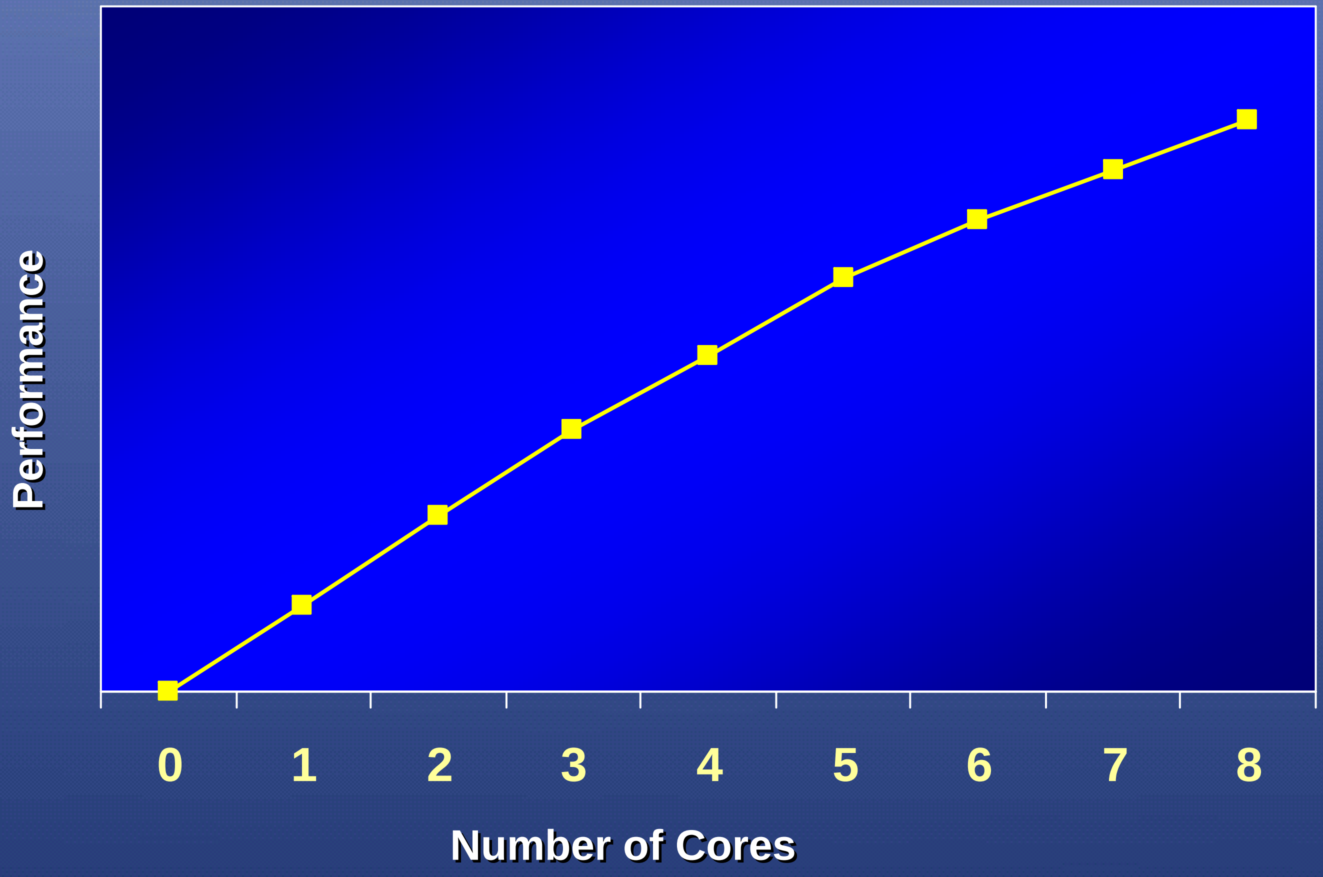
Benefits of FMN *



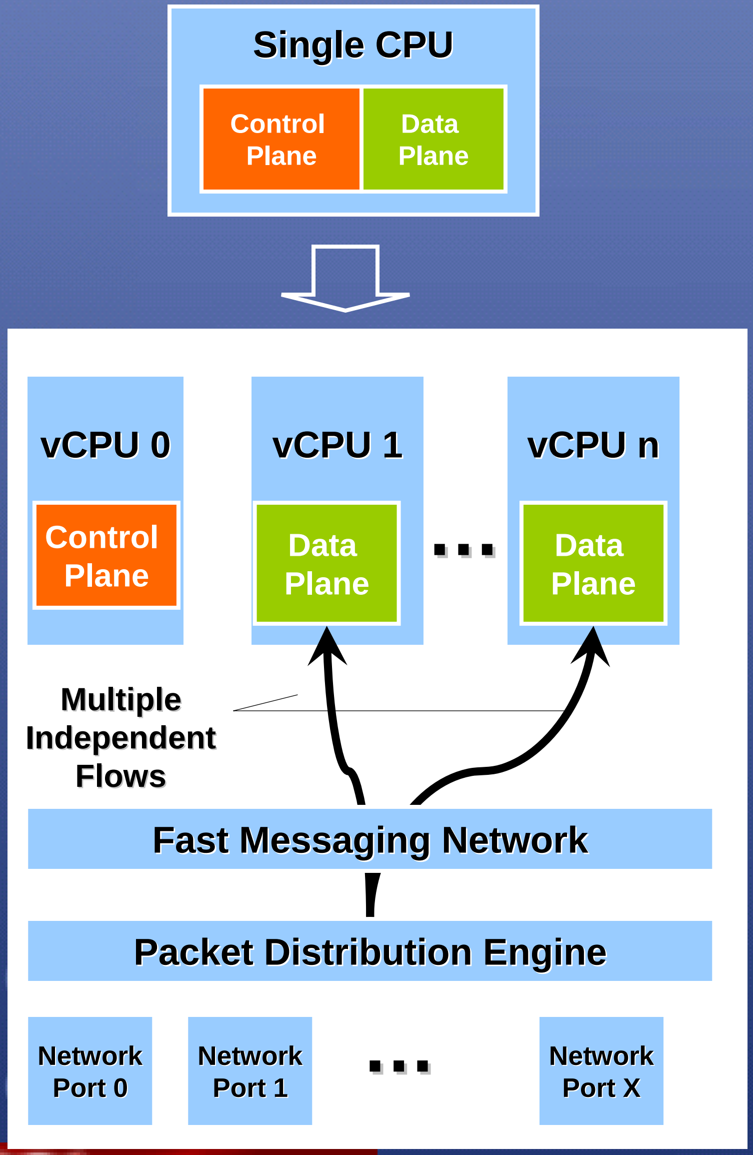
* FMN = Fast Messaging Network, unique to RMI architecture

- Enables linear scaling with number of CPU cores
  - Amdahl's law implies that locks limit performance regardless of CPU resources
  - FMN removes locks associated with device drivers
- Preserves CPU cycles for application processing, rather than interfacing with peripherals
  - Traditional NIC card requires CPU to spend a lot of time on driver code to access descriptors - up to 100 instructions
  - FMN designed to tightly integrated with software.
- Further improves performance by saving memory accesses and reducing memory footprint
  - CPU read/write message queues work without any memory access
  - Result: reduced pressure on the memory and cache subsystem, *and*
  - Increased application performance by minimizing overall memory latency

Performance Scaling with Multiple Cores

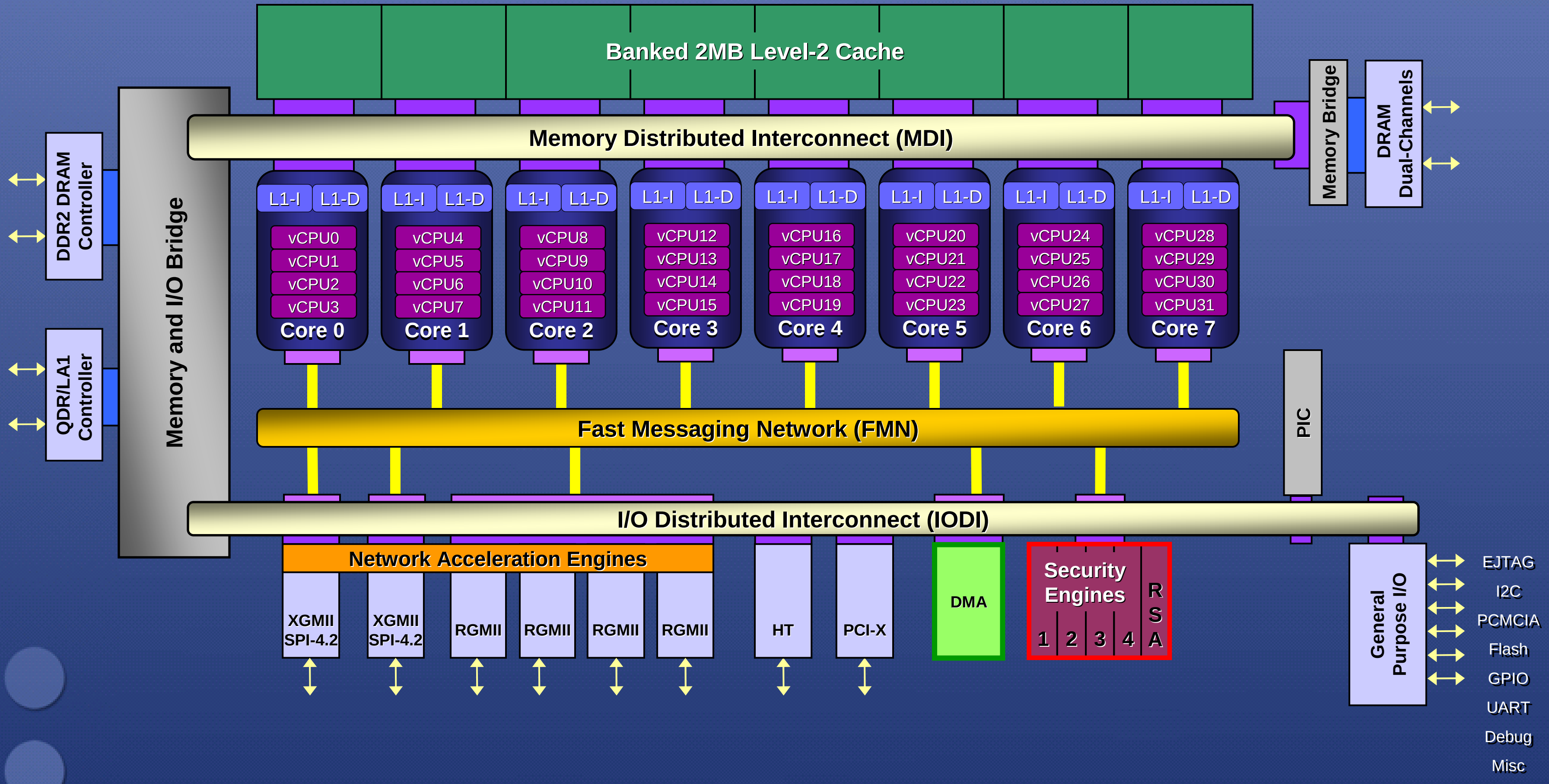


Straightforward Approaches to Migration

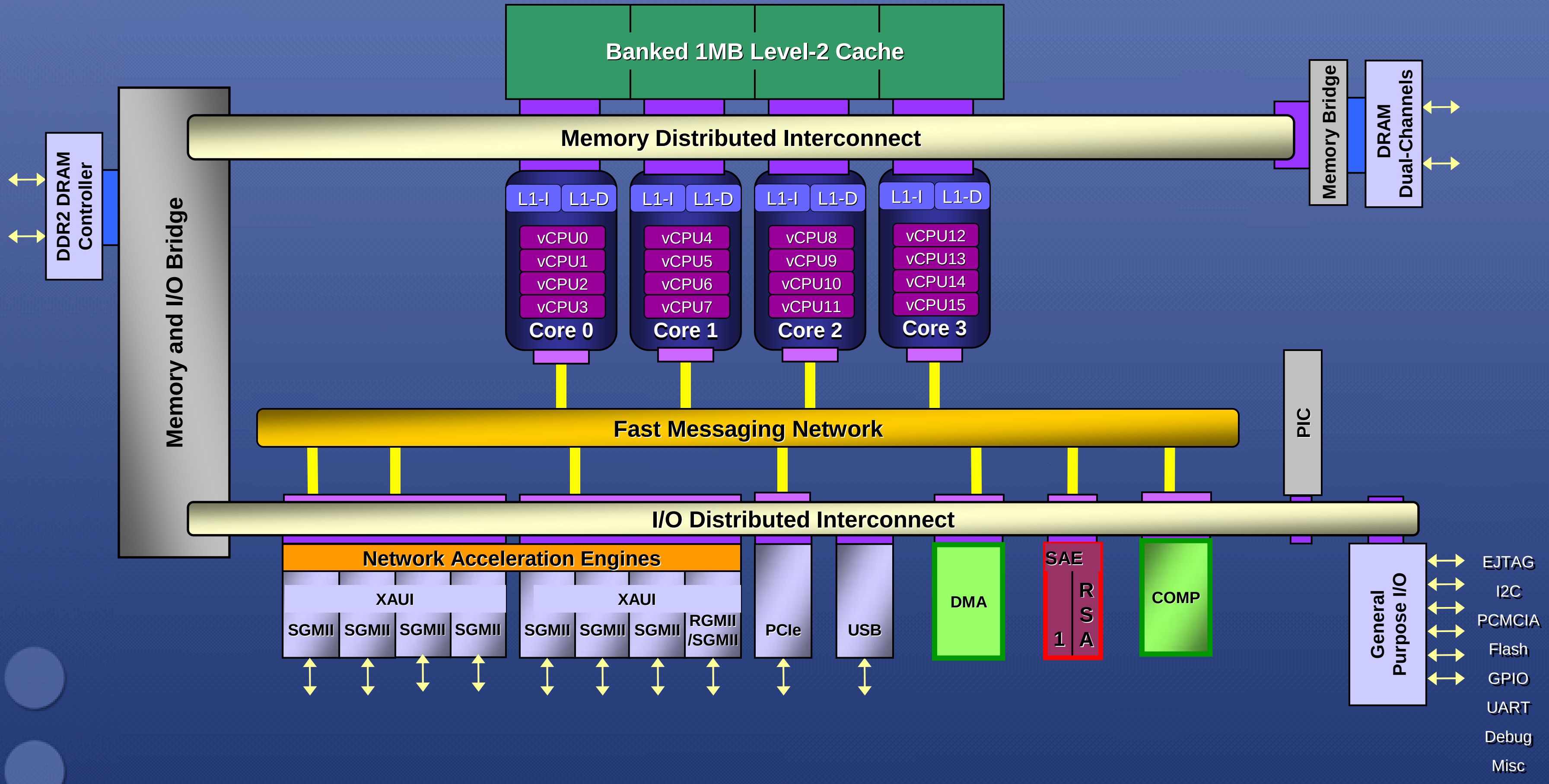


- SMP OS's automatically take advantage of multiple vCPUs.
- RMI architecture features FMN/CMS together with Intelligent Network Accelerator to distribute workload to multiple vCPUs
- Control path and Data path are easily partitioned and managed
- Multiple data planes run on an independent set of flows or pipelined no needs for synchronization

XLR® Processor Architecture



XLS® Processor Architecture

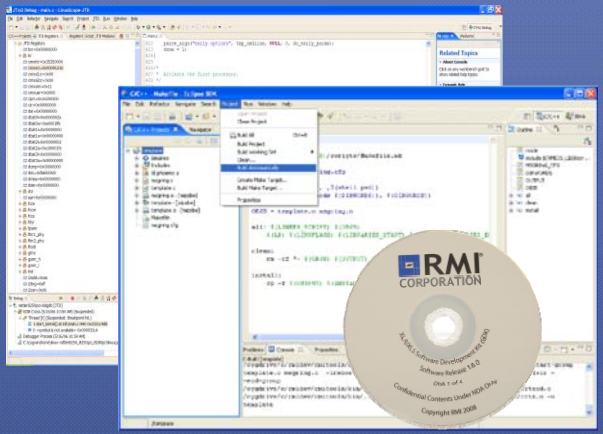


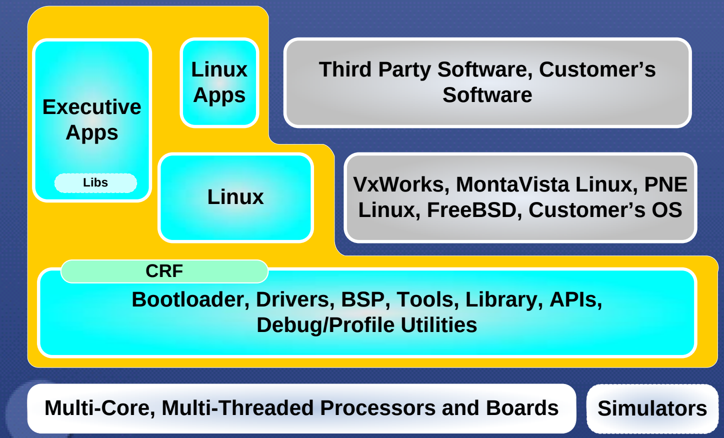
Conclusion

- RMI’s architecture addresses the critical issues of multi-core
  - Software migration from single- to multi-core
  - Meeting the market demand of high performance
  - Achieving system level scaling
- Many companies worldwide have achieved superb results by developing with RMI and plan to continue with our next generation

Software Infrastructure and FreeBSD Port

Next Generation, Multi-core, Multi-threading Programming Solutions Development Kits



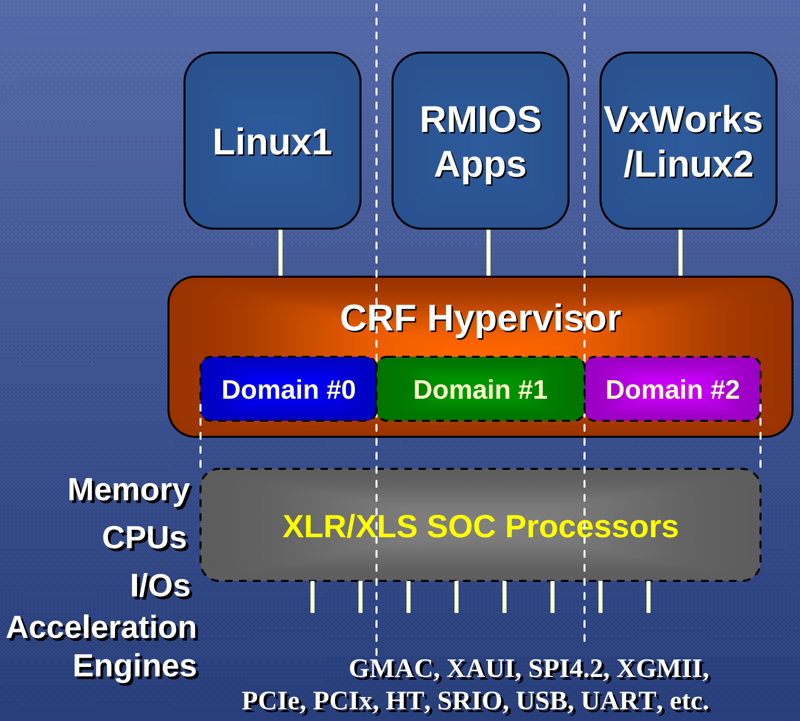

Executive Apps
Libs
Linux Apps
Third Party Software, Customer's Software
Linux
VxWorks, MontaVista Linux, PNE Linux, FreeBSD, Customer's OS
CRF
Bootloader, Drivers, BSP, Tools, Library, APIs, Debug/Profile Utilities
Multi-Core, Multi-Threaded Processors and Boards
Simulators

- Comprehensive Threading-Enhanced Debugger, Profiler and Analysis Tools
- Support for Multiple Industry Standard Operating Systems
- Multi-core, Multi-threaded, Multi-OS resource partitioning framework
- Reference Application Software
- All included in a complete Software Developer Kit

Chip Resource Framework (CRF)

- CRF is a thin software layer multi-core resource management hypervisor framework
- CRF is an infrastructure solution of Multicore software programming, brings up Multiple OSs (e.g. 2 or more Linux OSs) on RMI Multicore processor.
  - Dynamically change resource allocation on different domains
  - Dynamically start/stop/delete any domain without affects others.
  - Enables the critical resource sharing across different domains
- Does not impair system performance
- Uniquely addresses many issues involved in multi-core software
  - Manages chip resource: memory, network and system
  - Virtualizes chip resources: interfaces, messaging, consoles, and events
  - Aids implementation of alternative software architectures
  - Virtual Consoles and Event Queues
  - Enhances debug

Chip Resource Allocation, Management and Partitioning!

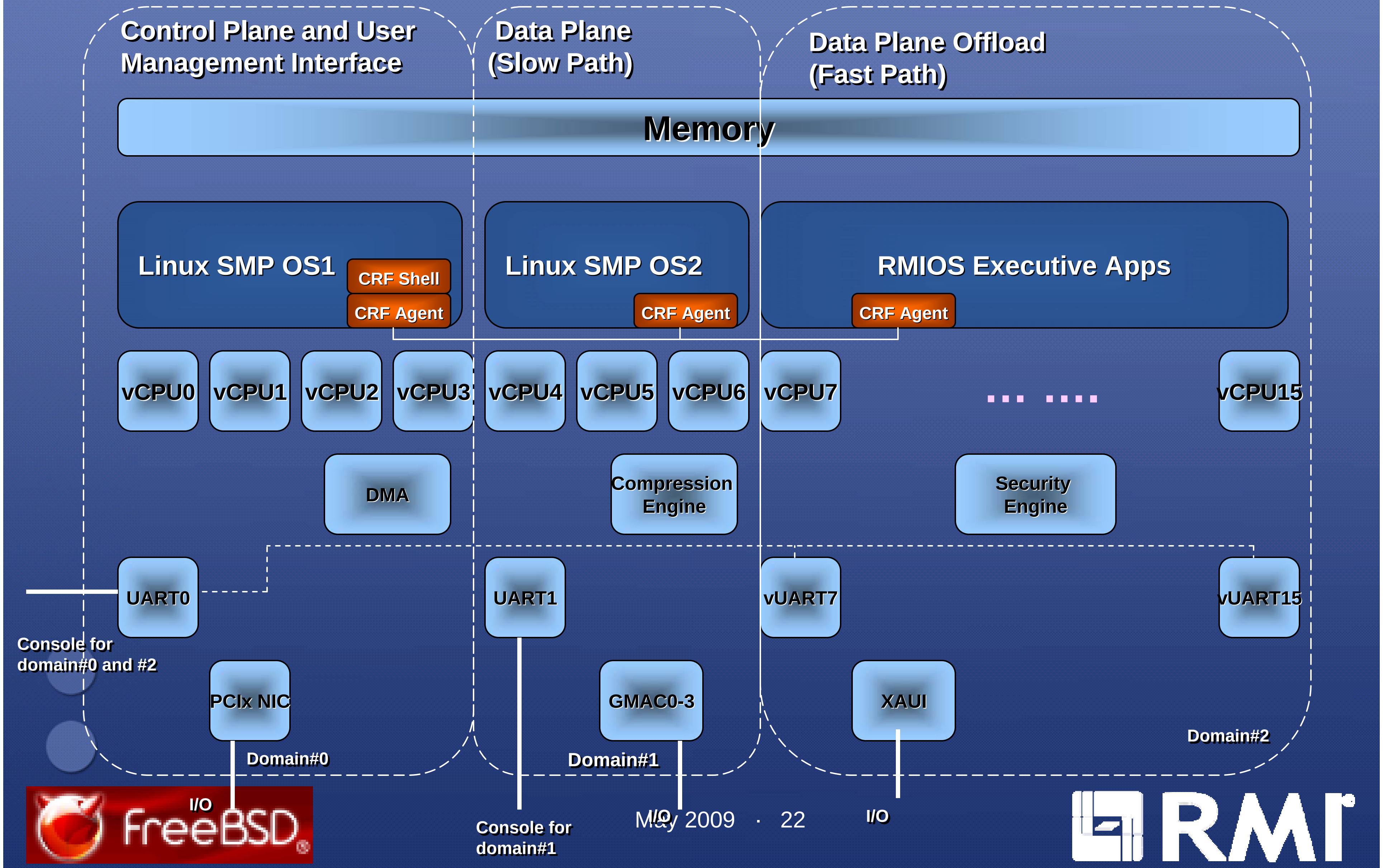


CRF solves the problems associated with the managing and running of multiple operating systems on a single chip!

FreeBSD

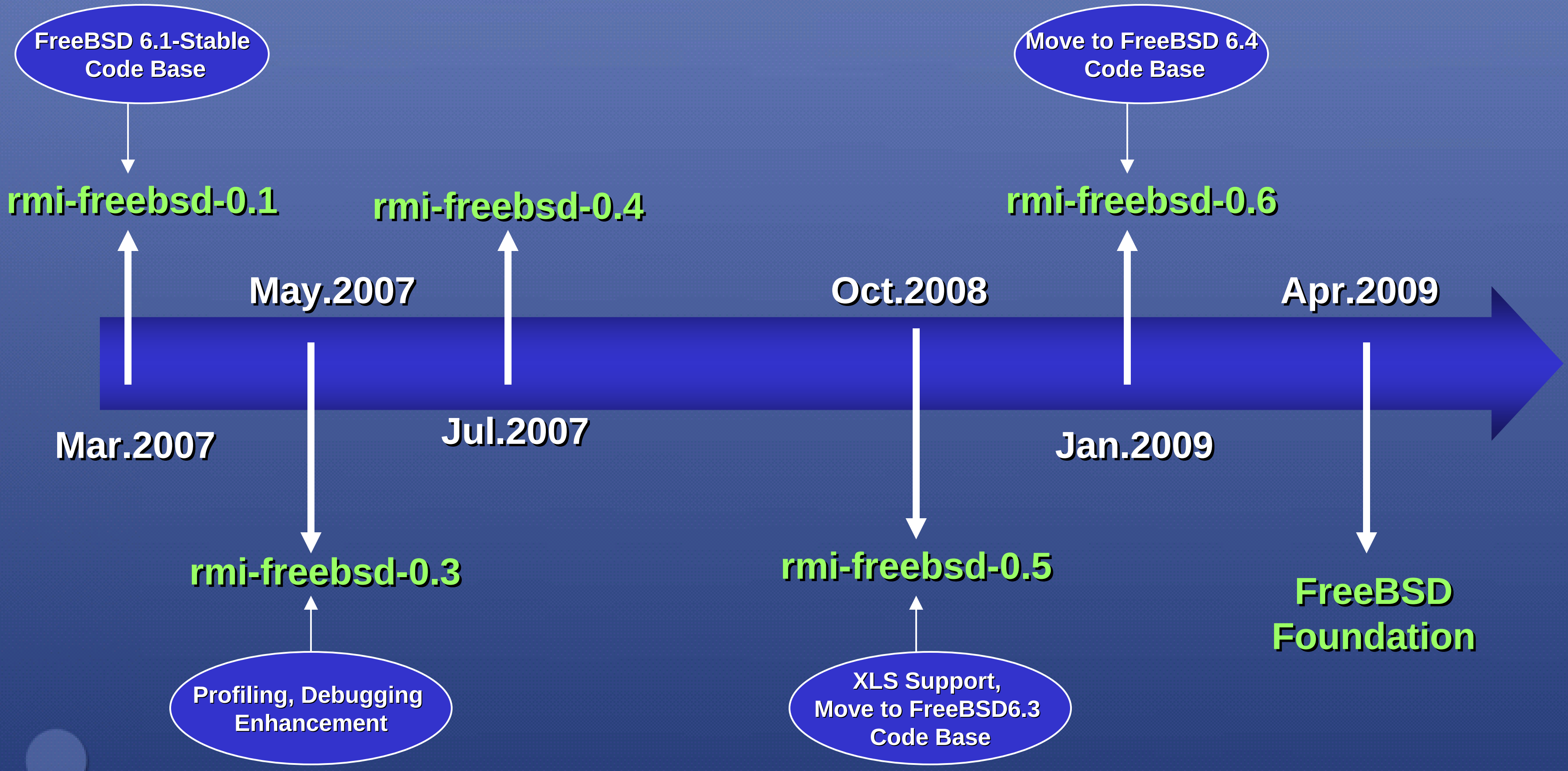
RMI

CRF Multiple OSs Solution Example



RMI

RMI FreeBSD Development



Contact

- George Jones (Vice President)
  - gjones@rmicorp.com

- Harrison Zou (Solutions Architect)
  - hzou@rmicorp.com

Thanks!