Systems Software

Ron Minnich, Mike Lang, Jim Garlick, Scott Pakin, Chris Dunlap, Pete Beckman, Jim Costa, Jim Laros, Pete Beckman, Sriram Swaminarayan, Costin Iancu, Ron Brightwell, Sue Kelly

SysSW Group Description

System Software Scope:

kernel, communication libraries, job scheduler, systems health and systems monitoring (RAS), security model, (not compilers)

We depend on:

- Hardware – we are the abstraction layer of the hardware
- Legacy considerations – old stuff has to work.

Who depends on us:

- Everyone else
- Systems Software presents a “Unix virtual machine” to apps
- Certainly on the Cray, but even on Blue Gene
- Desired interface converging to Linux API over time
  - see yesterdays “from laptop to exascale” bullet

Current Status

What will work

- **Petascale islands**
- **Hardware components based on COTS technology**
  - **E.g. port to BG/Q or Cray x86 is in some sense "done"**
- **Internode communication**

What won't work

- **Resilience**
- **Incremental Solutions – can't do current wisdom++**
- **Inter-core messaging and synchronization**
- **Dynamic Resource management for power, BW, locality, resilience, etc**
  - **But everyone we talked to wants a highly dynamic environment**

Exascale Challenges

- **SSW is itself becoming an application that needs resilience, is highly dynamic, and provides a rich set of composable services**
  - **Service set is app-dependent, and some services can exit once done (e.g. code that supports memory allocation or file loading)**
  - **Suggests that Linux monolithic kernel model is past its time**
- **Co-design with hardware critical to provide foundation**
  - **Hardware "resilience" can make software resilience *harder***
  - **Hardware "all fall down" model might not be what we want**
- **Providing**
  - **abstractions for Tools, VDA, PM, to interact with given a highly dynamic system environment**
  - **resilience services for various fault models, e.g. micro-checkpoints, software transactional memory, fault oblivious**
- **Lack of resources to develop and explore system software**
  - **Particularly testbeds**

Path Forward: Resilience

- Exploration of fault models – there are many candidates and no clear fault models
- Start multiple “explorations” into different fault models and run on testbeds/macroscale simulators
- Timeline (Phase I, II; Revolutionary)
- Required Funding: $20M
- Need to partner with some X-stack efforts, as well as programming models, as well as vendors to understand hardware fault model
  - It is almost certain failure unit will be “blob”, where “blob” is a mezzanine card/backplane/rack, unlikely it will be a core/cpu
  - And it might be the resilience hardware itself, as we saw 2 days ago
- Risk: risk aversion is a risk

Path Forward: New OS kernels for highly dynamic environments

- Exploration of virtualization and micro kernels
  - A la IBM efforts in KittyHawk, L4; new Bell Labs work on Osprey
- Start multiple "explorations", run on testbeds
- Timeline (Phase I, II; Revolutionary)
- Required Funding: $20M
- Need to partner with resilience to ensure it supports them
- Need to ensure that dynamism needs are supported
- "On-node service models" should be supported
  - E.g. coupled computing elements as publish/subscribe services
- Risks
  - Staffing: "Come work on your grandfather's code" is not a draw
  - But "Come work on something no one has ever done" – well, might work
  - As long as youtube/gmail/pandora are not blocked …

Path Forward: Power and Power API

- Vendor interaction to support global measurement and control
  - power equivalent of PAPI
  - Programming model interaction
- Timeline (Phase I; Evolutionary)
- Required Funding: $10M
- Need to talk with vendors about what we can learn
- COTS systems have an emphasis on local control and “automagic” behavior – quite opposite to what HPC people want
  - Need to know but decisions must be global, not local
  - “Think locally, act globally”
- Risk: unmanaged expectations
  - No, you can not turn off memory banks
  - Need a vendor talk: “Why you can’t get what you don’t want”

Path Forward: Memory Management Mechanisms (intra and inter)

- How do we share data, expose address space
  - Is that even the right question?
  - Is the real question: how do we efficiently communicate information?
  - How we're asking the question is conditioning the answer
  - Excessive memory sharing is a very bad idea on some NUMA systems
  - Calling it NVRAM may be a mistake; perhaps it should be called Less Volatile Ram (LVRAM)
- Programming model/hardware interaction
- Timeline (Phase I, II ; either Evolutionary or Revolutionary)
- Required Funding: $20M
- Another hardware/programming model interaction
- Risk: We might design a system that future hardware won't provide as it is too far away from what cell phones need
  - It's not PCs any more – why I invited Marvell

Path Forward: Dynamic allocation and reconfiguration

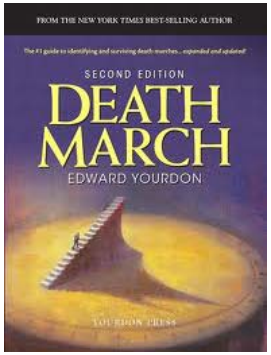
- Need a highly dynamic environment in which procs start up on nodes, attach to other procs, probe their state
  - Useful for debugging, analysis, viz
- How do procs export information and access via something a bit better than ptrace?
  - publish/use *named services*
- Need point design studies on testbeds
  - Can build on FAST-OS work
- Timeline (Phase II ; Revolutionary)
- Required Funding: $20M
- Partner with Tools and VDA
- Risk: we don't build on what we did on FAST-OS – since at least some of us addressed these very issues

Path Forward: Portable "API" – both up and down

- Robust abstractions to cover vendor-specific IP
  - GASNet as an example
- Point studies on testbeds
  - Current assumption is API is a system call
  - What about OS-per-core and high performance IPC a la Barrelfish
  - What about hardware FIFOs to talk to "kernel cores"
  - What about virtual machines and data channels a la L4/Osprey
  - What about "composable" kernels
  - Lots of good work but very little uptake in HPC community
- Timeline (Phase I & II; Evolutionary or Revolutionary)
- Required Funding: $10M
  - Can build on some FAST-OS and HPCS OS work
- Tight coupling to programming models and tools
- Risk is if we don't do this we miss the boat on the right API

Recommended Co-Design Strategy

- Critical steps/activities
  - Immediate interaction with hardware vendors
  - Defining communication/delineation with programming models
  - Testing new models/SSW interactions on petascale platforms and testbeds
- Working with vendors
  - Influence directly proportional to investment – pay to play
- Macro-scale simulation
- Role of skeleton/compact apps
  - With representative message traces from real runs
- Concerns/suggestions
  - System level testbed access
  - NNSA should field more testbeds
  - Not a technical problem

Big Picture Issues

- Coordination
  - Legal IP issues (e.g. Sandia can't collaborate w/ Google)
- Test beds
  - More access than just DST: Argonne as a model
  - Support O&M cost of testbeds: LANL as a model
- Simulators
  - Access to vendor hardware simulators: AMD as a model
- Remaining gaps
  - Progress requires more interaction from PM and other groups: hackathon!
  - Lack of well defined exascale hardware
- Other
  - heterogeneity
  - Finding a way to make NNSA labs exciting for young people