



# Future Directions in Cluster System Software

LCI's 5th International Conference, Linux Clusters: The HPC Revolution 2004

Wednesday May 19<sup>th</sup> 2004

**James H. Laros III**

Sandia National Labs

Scalable Systems Integration

*jhlaros@sandia.gov*



## Past

*(putting things in perspective)*

- Size
  - Large clusters 32, maybe 64 nodes
  - Sandia one of the few sites pushing the numbers higher
    - 256-512
- Run-time (OS, schedulers, *stuff that runs on the nodes*)
  - Most efforts not scalable
    - with the exception of Cplant™
    - GM, PBS... all pretty limited in scalability
    - Full blown Linux and other “stuff” running on nodes
- Cluster Management
  - Loosely coupled collection of scripts
    - Ours were a bit more tightly integrated but still left much to be desired.
  - Very cluster specific





## Present

*(Seems to be a few perspectives)*

- **Size**
  - Large clusters are ~512 nodes?
    - Certainly a few notable exceptions – see *the top500 list*
    - “Sweet spot” probably around 256
- **Run-time**
  - Scaling a bit further
    - Still not much integration between components
    - Still full Linux on disk-full nodes with lots of daemons
- **Cluster Management**
  - Most still have problems with scale
  - Lots of “packaging” disparate tools
  - Some good progress
  - Cluster Integration Toolkit (CIT)
    - Scalable, portable, extensible



## Future?

- **Trend: Nodes software stack has continued to get “heavier”**
  - Why? Can it get “lighter”?
- **Trend: Tools better but still don’t play well together**
  - Should be easy to integrate on a solid foundation
- **Question: Have software issues contributed to keeping Clusters smaller than we would have otherwise been able to utilize?**
- **Question: Are we missing something?**





## Light-os (for lack of a better name)

- **Started as an effort to simulate a light-weight kernel load**
- **Standard Linux Kernel source**
- **Process:**
  - **Create an initrd**
    - Busybox for utilities
    - Kernel modules for high-speed interface (etc)
    - Small amount of scripting
  - Tmpfs to act as final root file-system
  - Dump the excess baggage
- **Tested using alpha kernel source on 128 node system**
  - Could piggy-back the initrd onto the kernel image
  - initramfs with 2.6 should be able to do this?



## Light-os benefits

- **Disk-full**
  - “image” distribution
- **Disk-less**
  - No distribution but still have an “image”
- **No “image” necessary**
- **No more daemons**
  - Added stability - probably (less is more)
  - Stabilize run-times – probably (less interruptions)
  - What does run?
- **Very fast initializations**
- **Scales great**
- **It's just Linux!!**
  - No kernel source manipulation to maintain





## What is missing?

- How do we monitor the nodes health?
  - Intelligent Platform Management Interface (IPMI)
    - Lots of potential – spec (at least) good
    - “seems” like it is being adopted by vendors
      - Vendors taking too many liberties with the spec!
    - Provides a lot of what we need but not all (yet)
      - Mostly hardware status
- Scheduler, application allocation/distribution etc?
  - Cplant™ like solution?
- What about other cluster components?
  - Many components
  - Many different ways to obtain information
  - We need some glue



## RAS

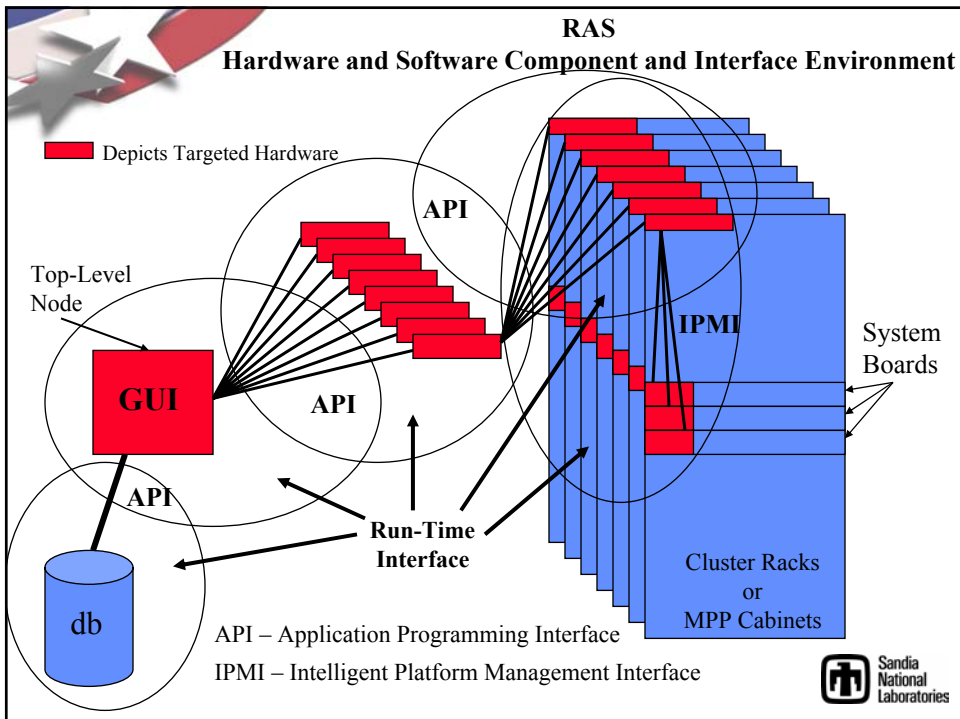
- Reliability – “The likelihood that a system or component will sustain full functional operation over its lifetime” – Typically measured in Mean-Time Between Failure (MTBF)
  - As component count increases calculation becomes more difficult.
  - What may be high MTBF’s for single components become unacceptable as component counts increase.
- Serviceability – “The measure of a system’s ability to sustain repairs to faulty components” – Typically measured in Mean-Time To Repair (MTTR) also “Time between service interruption and restoration”
  - You have to notice it failed first!!
  - Predictive replacement helps (also with MTBF)
  - Hot swap and redundancy helps (also with MTBF)



# RAS

cont.

- **Availability** – “The likelihood that a system is operational at any given time” – Typically measured in up-time percentage
  - In addition to a node being available we would like to have the greatest possible percentage of its resource available to the application.
  - Don’t waste cycles on “stuff”
    - Back to our motivations to “lighten” up the node





## RAS (cont.)

- The RAS “system” quickly becomes a small cluster of its own (for large cluster systems)
- LOTS of inference that hardware needs monitoring but what about software components?
  - MTBF and MTTR matters for software also!
    - Could be the hardest part!
- Could RAS be the “foundation” or “glue” that we have been talking about?



## RAS (cont.)

- New concept?
  - Definitely not.
  - We talk about RAS characteristics all the time.
  - Real RAS systems
    - ASCI Red
      - Probably the best example of what we are targeting.
      - RAS Will be an important part of Red Storm
- What can we leverage to make this happen?
  - Embedded technology again?
    - Probably be talking to embedded systems on devices
  - Leverage concepts from successful cluster management solutions
  - Use high end as an example





## Conclusions

- Lack of “flow” of technology from high end
  - We can learn a lot from the “higher-end”
- Much work to be done
  - Standards if well thought out and accepted can really help (if they remain “standard”)
- RAS work
  - Very do-able
  - Pretty large effort
- Interfaces between RAS, RunTime, Cluster Management
  - Challenging to get different efforts to play well together
  - If it was there would it be used?



## References

- CIT  
<http://www.cs.sandia.gov/cit>
- CPLANT™  
<http://www.cs.sandia.gov/cplant>
- RAS definitions taken from  
Sandia Report SAND2002-3164, Suzanne M. Kelly and  
Jeffery B. Ogden
- Computer Science Research Foundation (CSRF)  
<http://www.cs.sandia.gov/csri>
- Questions?
  - [jhlaros@sandia.gov](mailto:jhlaros@sandia.gov)

