



---

# **A Specialized Approach for HPC System Software**

**Ron Brightwell    Suzanne Kelly**

**Sandia National Laboratories**

**Arthur B. Maccabe**

**University of New Mexico**

**Sixth LCI International Conference on Linux Clusters**

**April 27, 2005**



# Outline

---

- **Brief history of Sandia/UNM lightweight kernels**
- **Reasons for a specialized approach**
- **Basic underlying principles**
- **Ongoing activities**
  - **Catamount development**
  - **Configurable OS research project**

# Sandia Experience

2004



## Red Storm

- ASC's next flagship
- 41 Tflops
- Custom interconnect
- Purpose built RAS
- Highly balanced and scalable
- Catamount lightweight kernel

1999



## Cplant

- Commodity-based supercomputer
- Hundreds of users
- Enhanced simulation capacity
- Linux-based OS licensed for commercialization

1997



## ASCI Red

- Production MPP
- Hundreds of users
- Red & Black partitions
- Improved interconnect
- High-fidelity coupled 3-D physics
- Puma/Cougar lightweight kernel

1993



## Paragon

- Tens of users
- First periods processing MPP
- World record performance
- Routine 3D simulations
- SUNMOS lightweight kernel

1990



## nCUBE2

- Sandia's first large MPP
- Achieved Gflops performance on applications



# Lightweight Kernel Goals

---

- **Targets high performance scientific and engineering applications on tightly coupled distributed memory architectures**
- **Scalable to tens of thousands of processors**
- **Enable fast message passing and execution**
- **Small memory footprint**
- **Persistent (fault tolerant)**



# Approach

---

- **Separate policy decision from policy enforcement**
- **Move resource management as close to application as possible**
- **Protect applications from each other**
- **Let user processes manage resources**
- **Get out of the way**



# Reasons for A Specialized Approach

---

- **Maximize available compute node resources**
  - **Maximize CPU cycles delivered to application**
    - **Minimize time taken away from application process**
    - **No daemons**
    - **No paging**
    - **Deterministic performance**
  - **Maximize memory given to application**
    - **Minimize the amount of memory used for message passing**
    - **Kernel size is static**
    - **Somewhat less important but still can be significant on large-scale systems**



# **Maximize Compute Node Resources (cont'd)**

---

- **Maximize memory bandwidth**
  - **Uses large page sizes to avoid TLB flushing**
- **Maximize network resources**
  - **Physically contiguous memory model**
  - **Simple address translation and validation**
    - **No NIC address mappings to manage**
- **Increase reliability**
  - **Relatively small amount of source code (~30K LOC)**
  - **Reduced complexity**
  - **Support for small number of devices**



## However....

---

- **Fixed memory regions may lead to inefficient use of memory**
  - Executable code that never gets used
  - Willing to waste memory to reduce complexity
- **Not all applications need large pages**
  - Considering adding run-time page size option



# Basic Principles

---

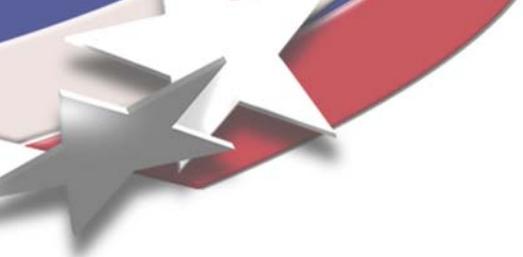
- **Logical partitioning of nodes**
- **Compute nodes should be independent**
  - **Communicate only when absolutely necessary**
- **Limit resource use as much as possible**
  - **Expose low-level details to the application-level**
  - **Move complexity to application-level libraries**
- **KISS**
  - **Massively parallel computing is inherently complex**
  - **Reduce and eliminate complexity wherever possible**



# Sandia Catamount Activities

---

- **Working on dual-core support**
  - Provides the ability to run in
    - Single-core mode (AKA heater mode)
    - Virtual node mode
- **Working on figuring out multi-core support**
  - Four dual-core nodes
- **OpenCatamount**
  - Open source version of Catamount
  - Working on general configure/build environment
  - Lacking one more piece of paperwork



**What's wrong with current operating systems?**



# Problems with General-Purpose OS's

---

- **Generality comes at the cost of performance for all applications**
- **Assume a generic architectural model**
  - **Difficult to expose novel features**
  - **Can't make everything look like an x86**



# Linux Memory Management

---

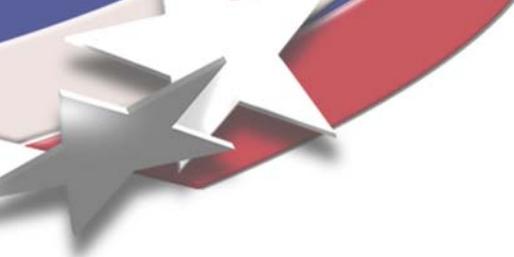
- **How much usable memory is there?**
- **Issues with page pinning**
  - **Linux wants to manage all of your pages for you**
  - **RDMA is a great model for high-performance networking**
    - **As long as a process' address map stays consistent**
  - **Linux memory management strategies are based on optimizing system performance by re-mapping memory pages frequently**
  - **Linux kernel developers don't like giving up control of resources**



## Various Quotes from the OpenIB List

---

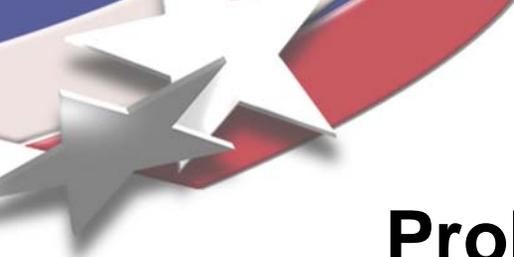
- **“If you take the hardline position that ‘the app is the only thing that matters’, your code is unlikely to get merged [into Linux]. Linux is a general-purpose OS.”**
- **“What doesn't work with that design [for page pinning] are the braindead designed-by-committee APIs in the RDMA world - but I don't think we should care about them too much.”**



## Interesting Side Note

---

- **Using GNU glibc increased the memory footprint of the “lightweight kernel” by 300% on Red Storm**



# Problems with Lightweight OS's

---

- **Limited functionality**
- **Difficult to add new features**
- **Designed to be used in the context of a specific usage model**

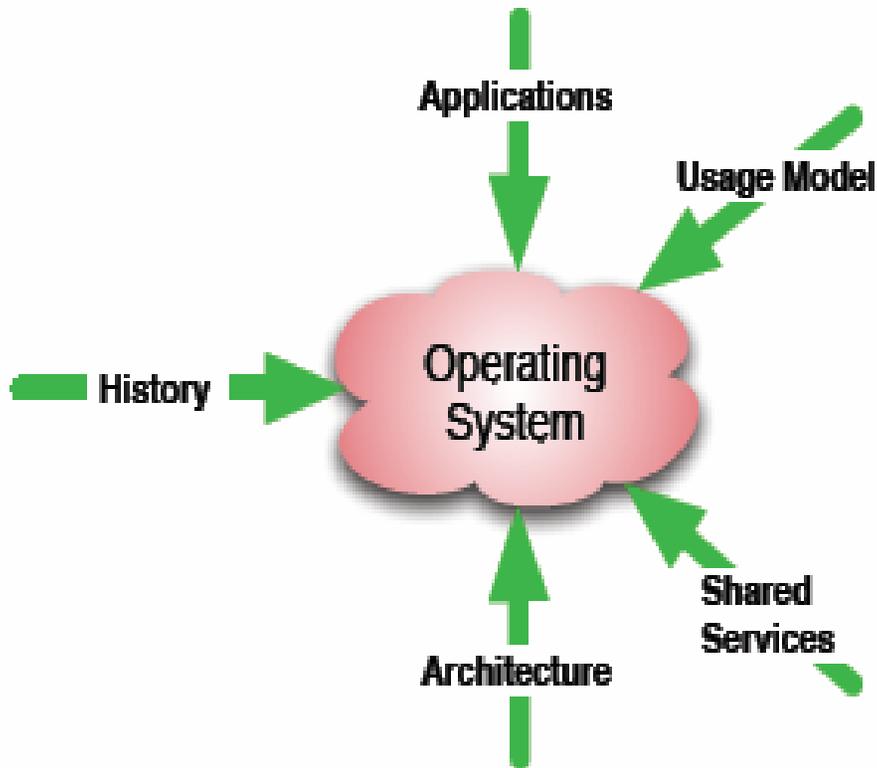


---

**The implementation and development of operating systems is an impediment to new architectures and programming models**

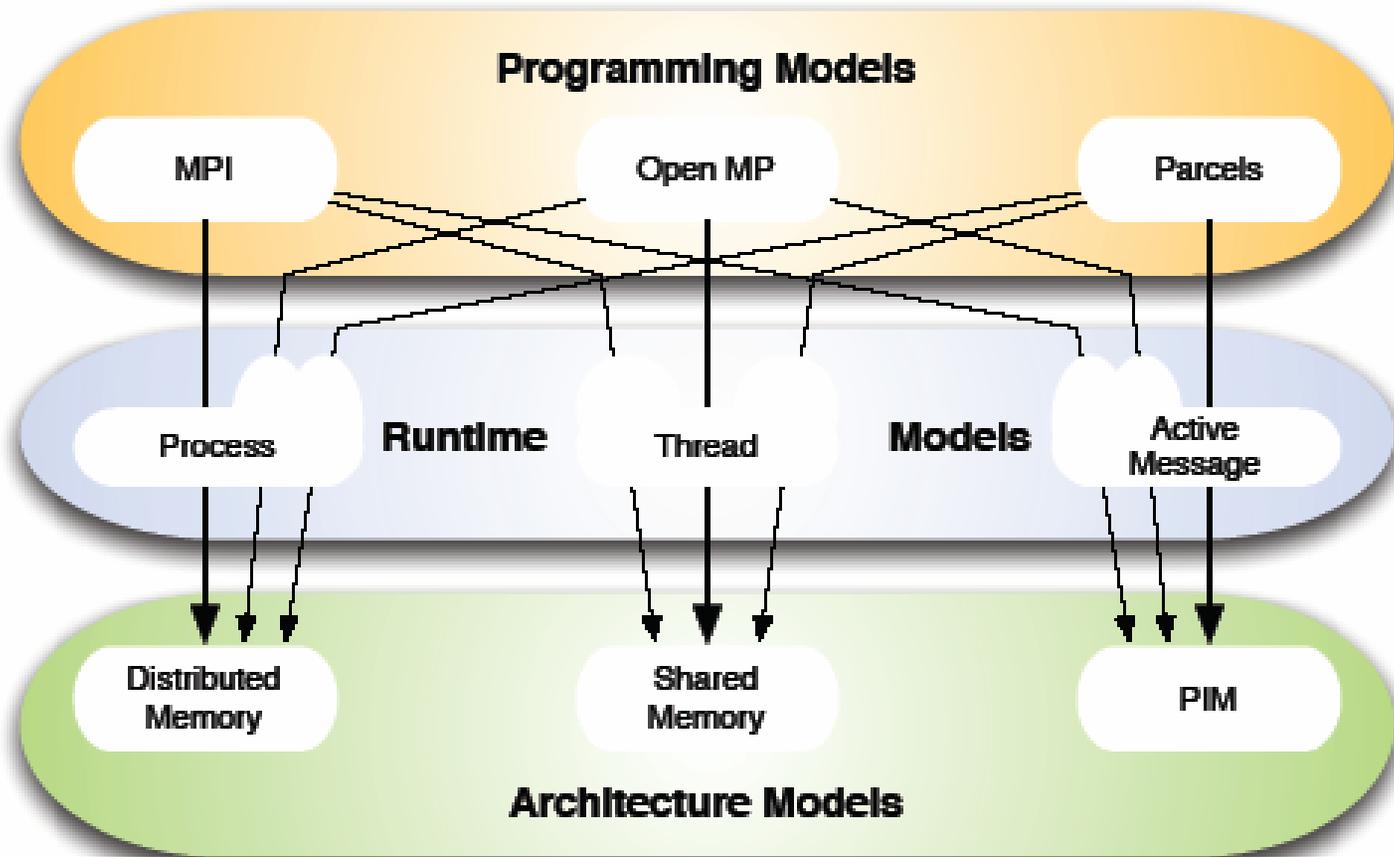
# Factors Influencing LWK Design

---

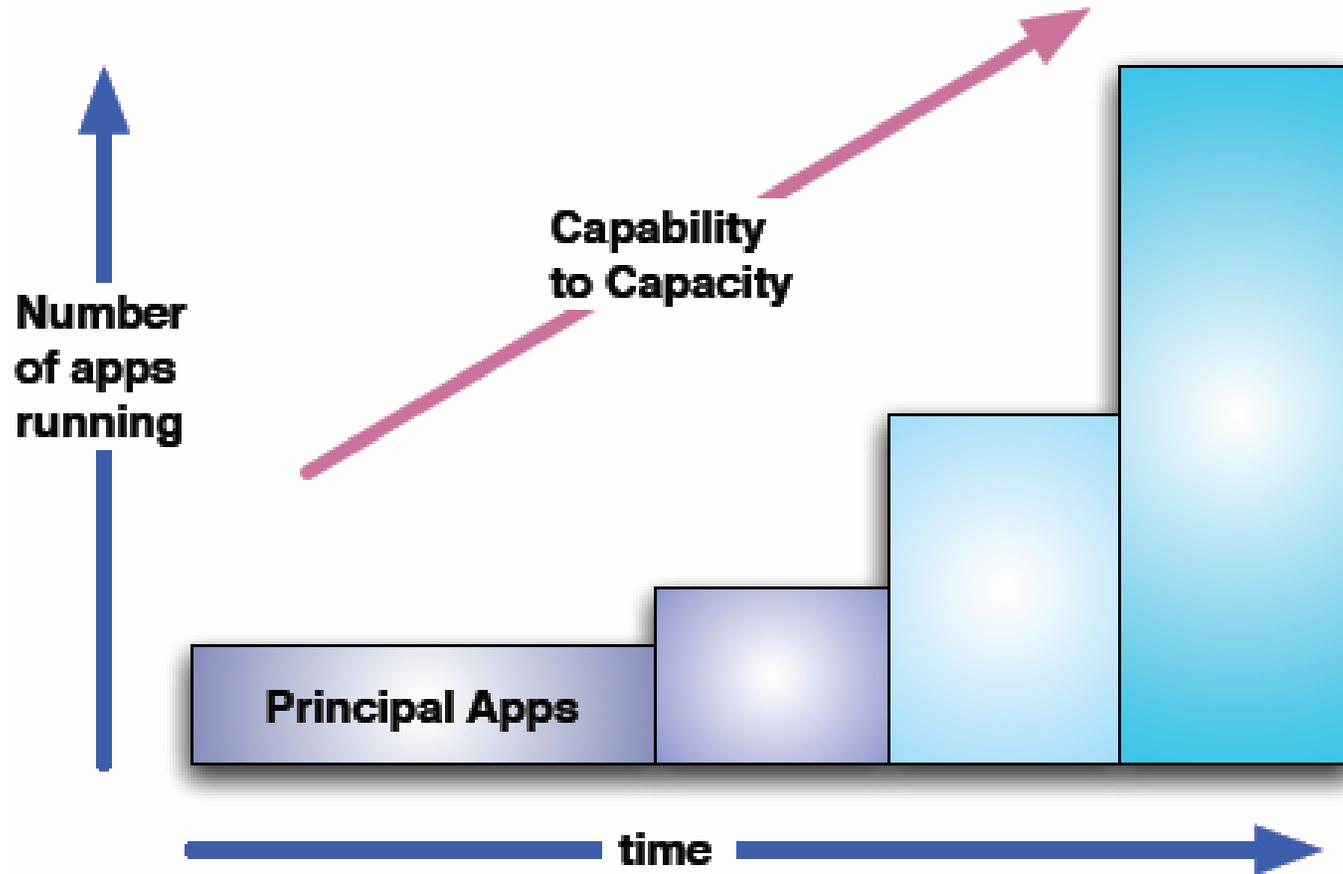


- **Lightweight OS**
  - Small collection of apps
    - Single programming model
  - Single architecture
  - Single usage model
  - Small set of shared services
  - No history
- **Puma/Cougar/Catamount**
  - MPI
  - Distributed memory
  - Space-shared
  - Parallel file system
  - Batch scheduler

# Programming Models



# Usage Models





# Current and Future System Demands

---

- **Architecture**
  - Modern ultrascale machines have widely varying system-level and node-level architectures
  - Future systems will have further hardware advances (e.g., multi-core chips, PIMs)
- **Programming model**
  - MPI, Thread, OpenMP, PGAS, ...
- **External services**
  - Parallel file systems, dynamic libraries, checkpoint/restart, ...
- **Usage model**
  - Single, large, long-running simulation
  - Parameter studies with thousands of single-processor, short-running jobs



# Configurable OS Project Goals

---

- **Realize a new generation of scalable, efficient, reliable, easy to use operating systems for a broad range of future ultrascale high-end computing systems based on both conventional and advanced hardware architectures and in support of diverse, current and emerging parallel programming models.**
- **Devise and implement a prototype system that provides a framework for automatically configuring and building lightweight operating and runtime system based on the requirements presented by an application, system usage model, system architecture, and the combined needs for shared services.**

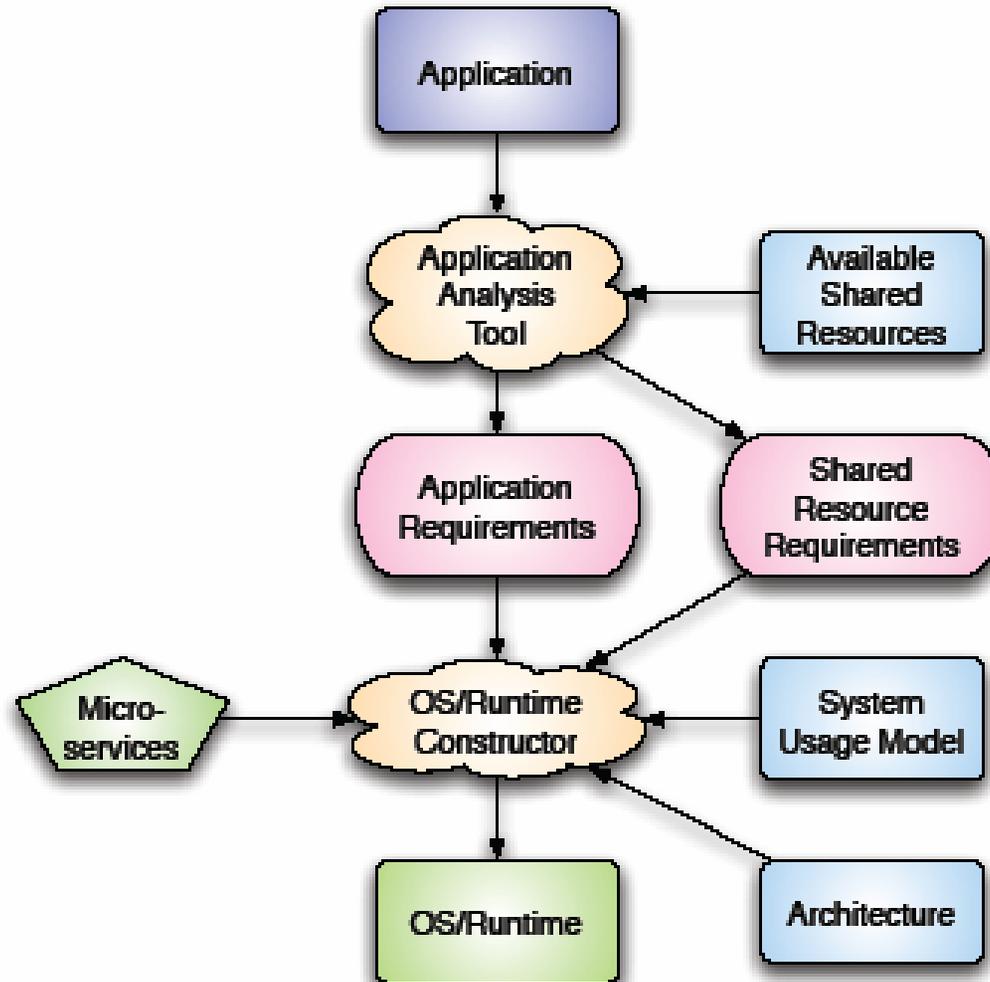


# Approach

---

- **Define and build a collection of micro-services**
  - **Small components with well-defined interfaces**
  - **Implement an indivisible portion of service semantics**
  - **Fundamental elements of composition and re-use**
- **Combine micro-services specifically for an application and a target platform**
- **Develop tools to facilitate the synthesis of required micro-services**

# Building Custom Operating/Runtime Systems





# Project Participants

---

- **Sandia National Laboratories**
  - Neil Pundit, Project Director
  - Ron Brightwell, Coordinating PI
  - Rolf Riesen
- **University of New Mexico**
  - Barney Maccabe, PI
  - Patrick Bridges
- **California Institute of Technology**
  - Thomas Sterling, PI
- **Project funded by DOE Office of Science under the FAST-OS program**