Linux emulation

Ron Minnich

Fifth IWP9
With thanks to Jim McKie

Outline

1. A quick overview of the Top 10 landscape
2. Our part in the landscape
3. The kernel part was the easy part
4. How we can run CNK binaries
5. Let's go look at code
6. Let's go look at machcnk
7. Conclusions

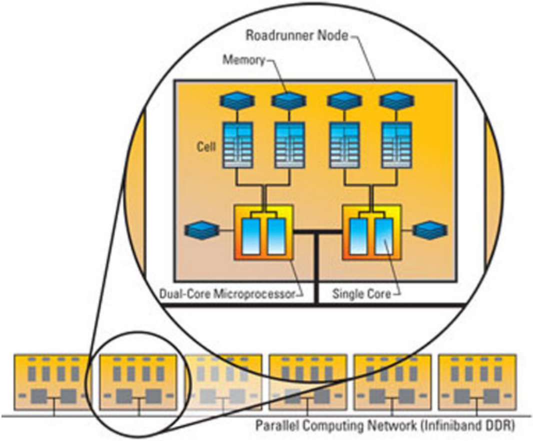
The Top 10

1. Jaguar - Cray XT5-HE Opteron Six Core 2.6 GHz
2. Nebulae - Dawning TC3600 Blade, Intel X5650, NVidia Tesla C2050 GPU
3. Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband\
4. Kraken XT5 - Cray XT5-HE Opteron Six Core 2.6 GHz
5. JUGENE - Blue Gene/P Solution
6. Pleiades - SGI Altix ICE 8200EX/8400EX, Xeon HT QC 3.0/Xeon Westmere 2.93 Ghz, Infiniband
7. Tianhe-1 - NUDT TH-1 Cluster, Xeon E5540/E5450, ATI Radeon HD 4870 2, Infiniband
8. BlueGene/L - eServer Blue Gene Solution
9. Intrepid - Blue Gene/P Solution
10. Red Sky - Sun Blade x6275, Xeon X55xx 2.93 Ghz, Infiniband

The top 10 by type

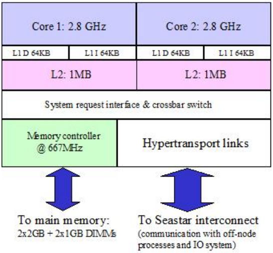
- These systems range in costs from $100M up
- No, it can't be done with Google clusters ...
- 5 COTS clusters @ 2, 3, 6, 7, 10
- 2 part-custom (Cray XT-5) @ 1, 4
- 3 full-custom (Blue Gene) @ 5,8,9

COTS Cluster Example: RoadRunner



Semi-Custom: Cray XT[4,5,6]

Current Dual Core Opteron



Core 1: 2.8 GHz | Core 2: 2.8 GHz

L1 D 64KB | L1 I 64KB | L1 D 64KB | L1 I 64KB

L2: 1MB | L2: 1MB

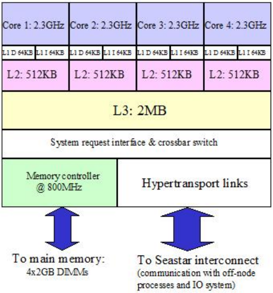
System request interface & crossbar switch

Memory controller @ 667MHz | Hypertransport links

To main memory: 2x2GB + 2x1GB DIMMs

To Seastar interconnect (communication with off-node processes and IO system)

Phase 2 Quad Core Opteron



Core 1: 2.3GHz | Core 2: 2.3GHz | Core 3: 2.3GHz | Core 4: 2.3GHz

L1 D 64KB | L1 I 64KB | L1 D 64KB | L1 I 64KB | L1 D 64KB | L1 I 64KB | L1 D 64KB | L1 I 64KB

L2: 512KB | L2: 512KB | L2: 512KB | L2: 512KB

L3: 2MB

System request interface & crossbar switch

Memory controller @ 800MHz | Hypertransport links

To main memory: 4x2GB DIMMs

To Seastar interconnect (communication with off-node processes and IO system)

Full-custom: Blue Gene/P CPU

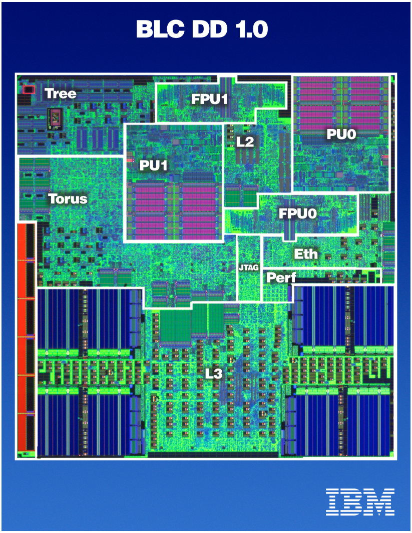


Figure: Blue Gene/L CPU

Blue Gene/P System

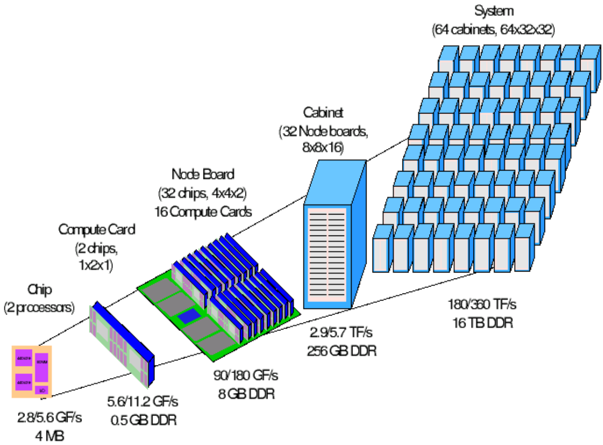


Figure 1: BlueGene/L packaging.

Figure: Blue Gene/L System

Background

- Over past four years HARE project (DOE, IBM, Bell Labs, Vita Nuova) has ported Plan 9 to two of the largest supercomputers in the world
- BG/L and then BG/P
- Research to study the value of Plan 9 in this context
- Plan 9 replaces two existing OSes: IBM's Compute Node Kernel (CNK) and/or Linux

Total port effort for BG/L

- 16 man weeks
- How much assembly in Plan 9 kernel? 1033 lines
- How many files in Plan 9 BG/L kernel? – About 90, including auto-generated by config
- 18 are platform-specific – – Of which we had to modify about 10
- Plan 9 (an OS) is smaller than every MPI library
- BG/P effort was similar
- You can see all our code: http://bitbucket.org/ericvh/hare

You think the kernel is big? you oughtta see the user-mode software!

- A very large system ... DCMF, a "simple" comms library, is 100KLOC of C++
- The *configuration file* for openmpi is 150KLOC
- And let us not forget the other runtime goo such as Python

You've go one million lines of C/C++/Fortran code for one program

- I'm including a lot of library code in this
- written in GNU C (its own standard) and GNU C++ (its own standard)
- Configured at run time with Python
- A source port to Plan 9 is simply not in the cards
- Just consider what it took just to port gcc – and it's not really working that well yet
- We considered several options, including source-source transformation, porting compilers, etc.
- None of these options was workable

Only practical option: run CNK binaries in Plan 9

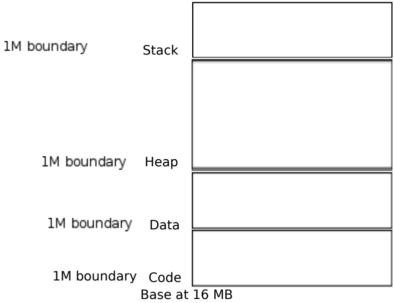
- Next question: how?
- First we need to see what a CNK binary looks like:
- 

Figure: A CNK Binary layout

- This suggests an idea
- If we want a Plan 9 "manager" program
- It can live in same address space as the binary itself

End up looking like this:

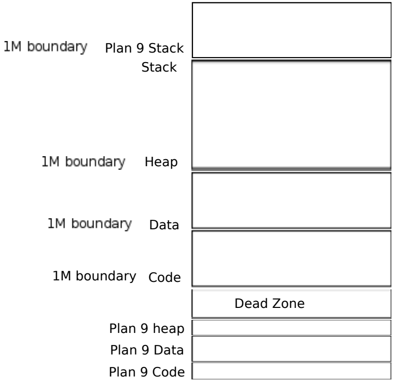


Figure: Plan 9 “shepherd” process image

Other issues ...

- In LinuxEMU, on x86, INT 80 is invalid and is trapped via notes to a manager
- Not practical on Blue Gene for several reasons
- Only one system call instruction used by Plan 9 and Linux and CNK
- Efficiency issue
- Kernel always knows more than user mode (e.g. it knows physical addresses and user only knows virtual)
- So it really only makes sense to trap in kernel

Trapping in kernel

- There's only one system call type
- So you have to mark the process and switch out on the mark
- OK, let's go look at some code

The code that runs the code

- Called machcnk since it runs libmach
- one interesting thing: you can flip back and forth
- Back into code ...

Conclusions

- We can emulate Linux in the Plan 9 kernel
- We can switch back and forth
- The "shepherd" model works on Blue Gene because we know where the binaries live
- On real Linux it's not nearly this easy
- Unless we recompile all the GNUbin to live above 16M – which is not that hard
- This is one path to efficient, integrated emulation in the Plan 9 kernel