HPC on Linux
Current Limitations and Futures

Roberto Innocente

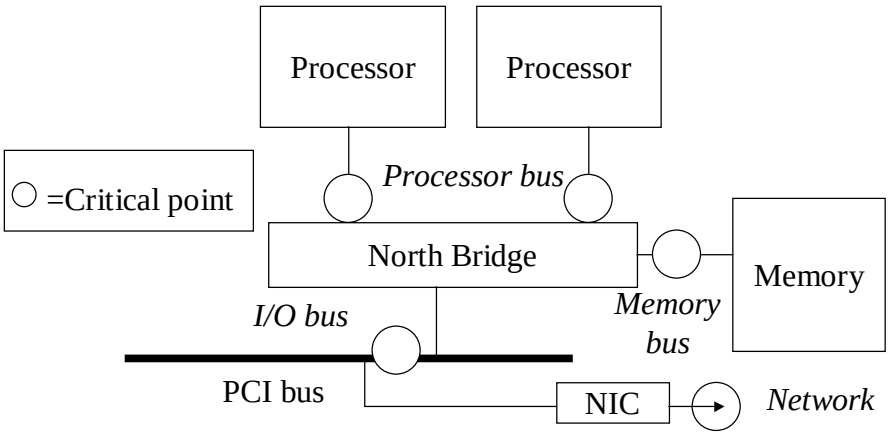
rinnocente@hotmail.com

- Hardware: *processor bus,I/O bus, memory bus,network*
- Software: *network layering,software layering,memory copies*

Hardware

Discussion on hardware will be mostly general, anyway the two most common architectures such as Alpha and IA32 will be explicitly covered.

Typical Node Architecture



Network (Physical Layer)

Current technology is at ~1 Gb/s(GbE,Myrinet), is there room for improvement?

- Ethernet 2.5 Gb/s ... 10 Gb/s
- Myrinet2000 2Gb/s
- SONET OC192 ...
- GSN(Hippi) 6.4 Gb/s

A lot of the improvements in the optical arena are coming from the use in the last years of the low cost VCSELs(Vertical Cavity Surface Emitting Laser)

PCI Bus

Standard PCI in use today is 32 bits at 33 Mhz, just sufficient for 1 Gb/s technologies,is there a path for better throughput?

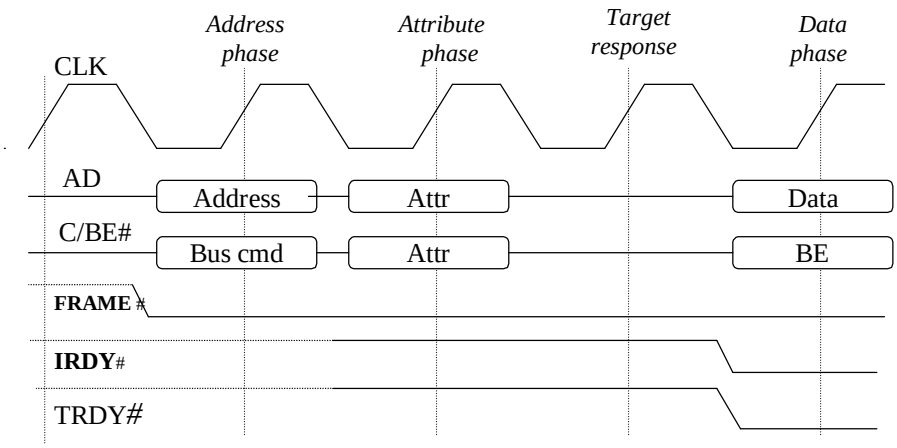
- PCI32/33 4 bytes@33Mhz=132MBytes/s (on i440BX,...)
- PCI64/33 8 bytes@33Mhz=264Mbytes/s
- PCI64/66 8 bytes@66Mhz=528Mbytes/s (on i840)
- PCI-X 8 bytes@133Mhz=1056Mbytes/s

PCI-X will implement split transactions

PCI efficiency

- Multimaster bus but arbitration is performed out of band
- Multiplexed but in burst mode (implicit addressing) only start address is txmitted
- Fairness guaranteed by MLT (Maximum Latency Timer)
- 3 / 4 cycles overhead on 64 data txfers < 5 %

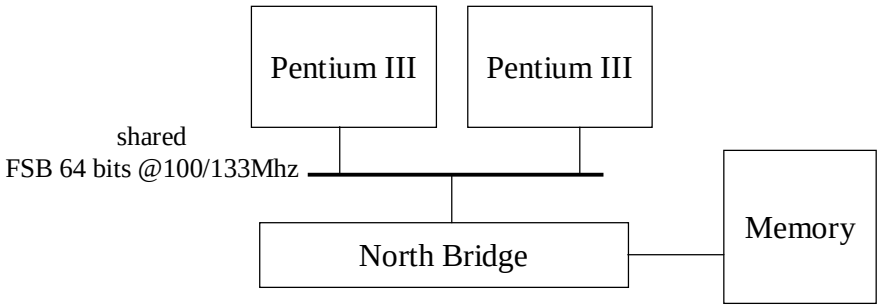
PCI 2.2/X timing diagram



Processor bus

- Intel (AGTL+):
  - bus based (max 5 loads)
  - explicit in band arbitration
  - short bursts (4 data txfers)
  - 8 bytes wide(64 bits), up to 133 Mhz
- Compaq Alpha (EV6):
  - point to point
  - licensed by AMD for the Athlon
  - source synchronous(up to 400 Mhz)
  - 8 bytes wide(64 bits)

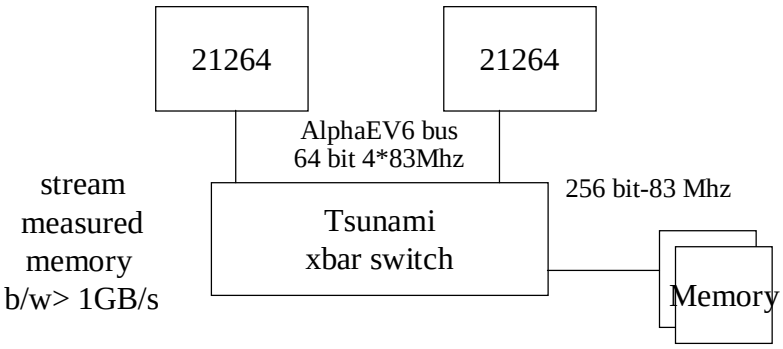
Intel IA32 node



Intel PIII processor bus

- Bus phases :
  - Arbitration: 2 or 3 clk
  - Request phase: 2 clks packet A, packet B (size)
  - Error phase: 2 clks, check parity on pkts,drive AERR
  - Snoop phase: variable length 1 ...
  - Response phase: 2 clk
  - Data phase : up to 32 bytes (4 clks, 1 cache line)
- 13 clks to txfer 32 bytes

Alpha node



Alpha EV6 bus

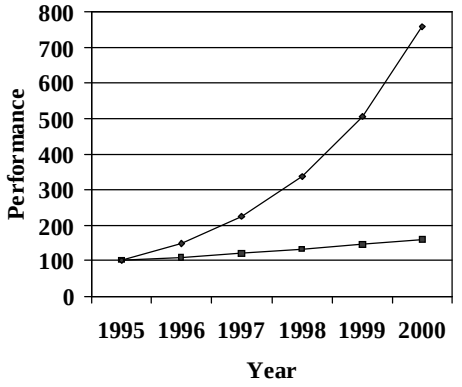
- 3 high speed channels :
  - Unidirectional processor request channel
  - Unidirectional snoop channel
  - 72-bit data channel (ECC)
- up to 400 Mhz (4 x 100 Mhz: quad pumped)

Pentium 4 (Willamette)

Current ia32 architecture has severe limitations due to its processor bus but...Pentium4 ...

- 1.4/1.5 Ghz with 256 KB L2 cache on-chip will be available next month (SKT 423)
- Processor bus at 100 Mhz but Quad pumped
  (2x address rate/4x data rate)
- On sample mobos at 1.2 Ghz streams gives ~ 1.2 GB/s memory bandwidth
- Cache line will be sectored 64 bytes/ L3 cache up to 4MB/ L2 up to 1MB

Processor/Memory performance



CPU ~50%/year

Memory < 10%/year

Memory buses

Current measured memory b/w is low(i386 <400 MB/s) or medium(alpha ~1 GB/s), what can we do?

- SDRAM 8 bytes wide (64 bits)
  - PC-100 PC-133
  - DDR PC-200, QDR on the horizon
- RDRAM 2 bytes wide(16 bits)
  - RDRAM 600/800 double data rate

Memory bandwidth can easily be improved trough parallelism (alpha tsunami chip has 2x SDRAM banks), RDRAM and/or QDR (Quad data rate),but on i386 the current limiting factor is the processor bus

NIC Interconnection point

(from D.Culler)

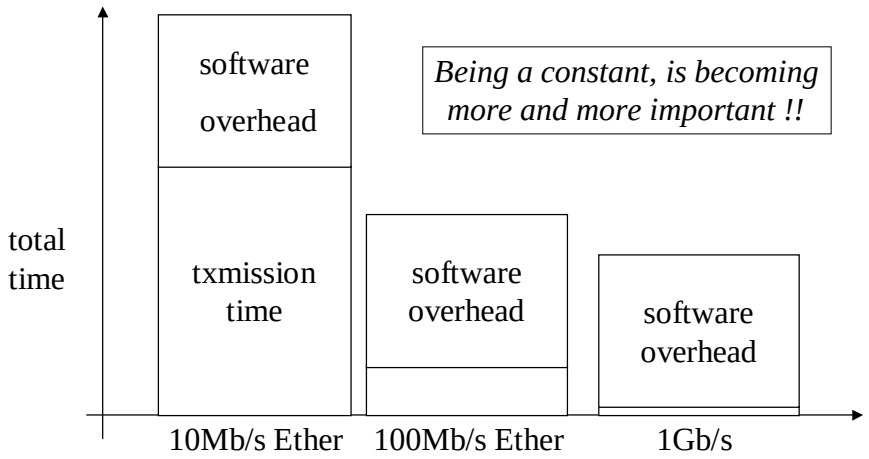
	Controller	Special uproc	General uproc
Register	TMC CM-5		
Memory	T3E annex	Meiko CS-2	Intel Paragon
Graphics Bus	HP Medusa		
I/O Bus	Many ether cards	Myrinet, 3ComGbe	SP2, Fore ATM cards

Software

Despite great advances in network technology(2-3 orders of magnitude), much communication s/w remained almost unchanged for many years (e.g.BSD networking).

There is a lot of ongoing research on this theme and very different solutions are proposed(zero-copy, page remapping,VIA,...)

Software overhead



Being a constant, is becoming more and more important !!

Zero Copy Research

High speed networks, I/O systems and memory have comparable bandwidths -> it is essential to avoid any unnecessary copy of data !

- Shared memory between user/kernel:
  - Fbufs*(Druschel,1993)*
  - I/O-Lite *(Druschel,1999)*
- Page remapping with copy on write *(Chu,1996)*
- Blast: hardware splits headers from data*(Carter,O'Malley,1990)*
- Ulni (User-level Network Interface): implementation of communication s/w inside libraries in user space

OS bypass – User level networking

- Active Messages (AM) – *von Eicken, Culler (1992)*
- U-Net –*von Eicken, Basu, Vogels (1995)*
- PM – *Tezuka, Hori, Ishikawa, Sato (1997)*
- Illinois FastMessages (FM) – *Pakin, Karamcheti, Chien (1997)*
- Virtual Interface Architecture (VIA) – *Compaq,Intel,Microsoft (1998)*

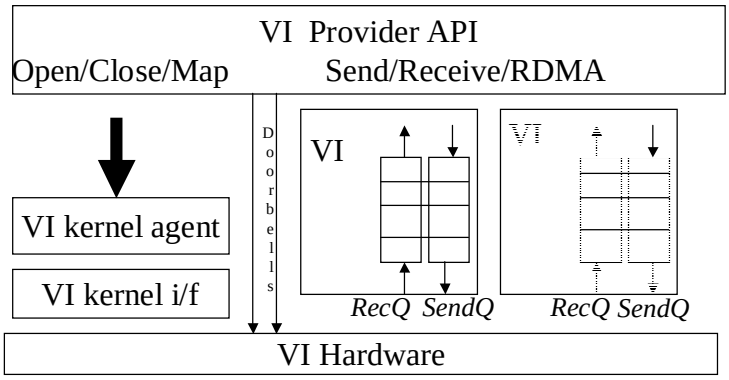
Active Messages (AM)

- *1-sided* communication paradigm(no receive op)
- each message as soon as received triggers a *receive handler* that acts as a separate thread (in current implementations it is sender based)

FastMessages (FM)

- FM_send(*dest,handler,buf,size)*
  sends a long message
- FM_send_4(*dest,handler,i0,i1,i2,i3)*
  sends a 4 words msg (reg to reg)
- FM_extract()
  process a received msg

Virtual Interface Arch. (VIA)

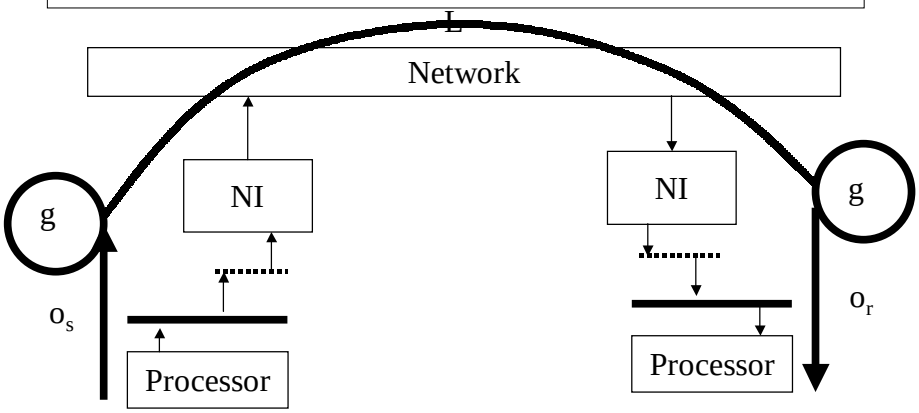


LogP metrics (Culler)

This metric was introduced to characterize a distributed system with its most important parameters,a bit outdated, but still useful.(e.g..does'nt take into account pipelining)

- L = Latency: time data is on flight between the 2 nodes
- o = overhead: time during which the processor is engaged in sending or receiving
- g = gap : minimum time interval between consecutive message txmissions(or receptions)
- P = # of Processors

LogP diagram



time=o_s+L+o_r

Software layering

Use of abstraction layers has promoted generality, but maybe it can be harmful to efficiency

A typical read/write on a tcp socket passes trough:

- VFS(Virtual File System) layer
- BSD socket layer
- Inet socket layer

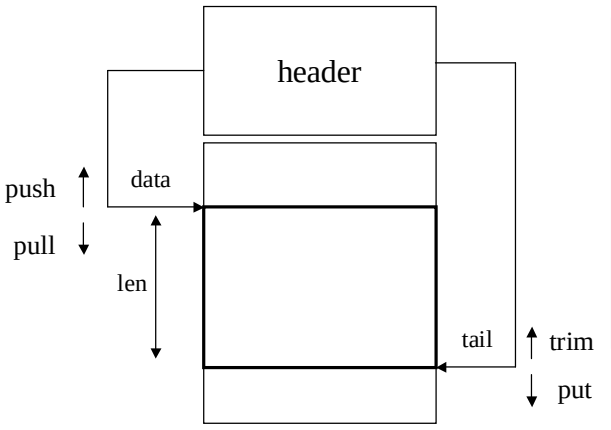
Network layering considered harmful ?

Is the successful network layering approach to networking harmful to today high speed network performance ?

- 7 layers ISO/OSI model
- 4 layers TCP/IP

Yes, if it implies data copying between layers, no if layering is just an abstraction

Linux Socket buffers (sk_buff)



This is the Linux way to avoid copying between network layers, does'nt avoid copies between kernel/user spaces and for frag/defrag-mentation