Clusters In Space

James Lux, P.E.
Jet Propulsion Laboratory
james.p.lux@jpl.nasa.gov
Space is a (really) different place to work and design for
Why do we need a cluster?

Do science that cannot be done without one!

New types of sensors produce huge data volumes (TB/day)

But, you’re a long way from Earth

Low data rates (8 bps to 128 kbps)

- On-board processing can do a lot to reduce the data volume
  - Spacecraft and spacecraft systems can’t use latest and greatest processors, so if you need lots of processing, you use lots of older, slower ones. (i.e. a cluster)
  - You get scalability and reliability “for free”
Critical Resources
Design Constraints

- Flight Resources
  - Mass
    - 10’s - 100 kg
  - Power
    - 10’s – 100’s of Watts
    - About 300 W/m²
    - Heat radiation
  - Volume
    - few liters
  - Data bandwidth
    - few kilobits/sec

- Aspects of Development
  - Long development time cycle
  - Risk management
  - Component limitations
    - Traceability
    - Preference for old, proven designs
  - Extensive testing
  - Bodies to work on it
Peculiar Environments

- Vacuum
  - it’s a really, really good thermal insulator
  - Radiation cooling to 3K cold space
- Thermal
  - High gradients (one side cold, the other hot)
- Radiation
  - Photons, electrons, protons, heavy ions, and neutrons, Oh My!
- Shock and Vibration
  - Launch, separation, and landing
- Electromagnetic
  - Can’t interfere, and you might be next to a big transmitter!

All these can, but remember the constraints!
Power, Mass, Volume, Risk
So what can you use to build a “cluster in space”

- Processors
  - Typically older processors that have versions ported to “spaceflight qualified fab” (8051, ADSP21020, RAD6000, RAD750, SPARC 7, 8)
  - Minimal cache

- Memory
  - Error Detection and Correction (e.g. 11 bits for 8)

- Mass Storage
  - RAM, Flash, EEPROM, PROM (no disk drives!)

- Interconnects
  - Wired
    - Serial point to point (good old RS-422)
    - MIL-1553 (dual bus, 1 Mbps)
    - Spacewire (IEEE-1355, 400 Mbps, point to point)
  - Optical w/cables
    - AS-1774 (high speed optical MIL-1553)
    - Roll-your-own
  - Optical Freespace
    - Very developmental right now, but great promise

- OS
  - Linux, VxWorks, Virtuoso, various RTOS
Put it all together

- A “breadboard” cluster in space starts to look a lot like early Beowulf clusters.
  - 80486 or Pentium scale processor with limited RAM
  - Point to point interconnects with multiple interfaces on each node
  - Diskless nodes
  - Stripped down operating system without many bells & whistles
- It looks a lot different than what people are building now, but,
- The concepts behind the design are the same, and familiar.
Two examples

- **Breadboard DSP Scatterometer processor**
  - Loosely coupled DSPs to do signal processing done by special purpose FFT chips and FPGAs
  - Established architecture is scalable to meet requirements & synchronization possible without working too hard
  - Established that you can use off the shelf flight qualified hardware to do the job (so the only NEW stuff is software)

- **Autonomously Controlled Element Phased Array**
  - Relies on cluster techniques (specifically distributed processing) for an infinitely scalable phased array
  - Overcomes limitations from relying on passive structural stiffness by measuring movement, predicting future behavior, and compensating in real-time
DSP Scatterometer Breadboard

Orbiting Instrument

Ocean Surface

Radar "footprint" divided into range slices

Repeat 400 times per second
How much work, and how to do it

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of Arithmetic Ops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noise filter</td>
<td>50k</td>
</tr>
<tr>
<td>Echo Dechirp</td>
<td>10k</td>
</tr>
<tr>
<td>Prefilter &amp; decimate</td>
<td>60k</td>
</tr>
<tr>
<td>FFT</td>
<td>21k</td>
</tr>
<tr>
<td>Mag &amp; Sum</td>
<td>2k</td>
</tr>
<tr>
<td>Total</td>
<td>143k ops/pulse</td>
</tr>
</tbody>
</table>

@ 212 pulses/second = 30.3 Mops/sec
(doubles for Polarimetric (2 channel) (60 Mops/sec))

Raw Arithmetic Load 60 MOPS
Implementation overhead (50%) 30
Subtotal 90 MOPS
Margin (100%) 90
Required Processor Power 180 MOPS

Flight DSP (TSC21020F) 40 MOPS
>> It’s going to take 5 processors to handle the load

Embarassingly Parallel?
Time/Frequency Synchronization

- Individual modules responsible for determining when to sample, and for sample rates
- Each module needs to know:
  - Absolute time – Own clock offset from “master”
  - Clock rate / Oscillator frequency
    - sample rate affects signal processing
    - clock rate affects whether local clock runs fast/slow
- Adapted version of Network Time Protocol (NTP) and GPS Disciplined clocks techniques
  - Periodic time message (delivery not critical)
    - “At the tone, the time is…”
  - Single, low rate, sync pulse (1 pulse/second) to all modules
  - “Byzantine Generals” algorithms can be used to handle unreliable message and tick transmission
Local Clock used for timing and control

- Each module “knows” its own local clock offset and rate
  - messages and periodic sync pulse
- Messages arrive saying, in effect:
  - Digitize and process an echo starting at absolute time 123377.4523825 seconds for a duration of 2.050 milliseconds
- Module turns that into local time and number of samples to digitize
  - in terms of “local clock ticks”

- Algorithms can “predict” ahead based on previous measurements of variations
  - Temperature
  - Aging
- Missing a 1pps tick or a sync message doesn’t have a huge effect.
Autonomous Controlled Element Phased Array

- Very large (>100 m) electronically scanned phased array for space based radar and communications applications.
  - Existing space based radar antennas use mechanical stiffness to hold their shape.
  - Limited spacecraft mass and volume means that you want lightweight, deployable antennas. Light, deployable ≠ stiff

- Concept is to continuously measure the shape of the antenna and adjust it to compensate.
  - Phased arrays are composed of thousands of essentially identical elements that are adjusted to create a desired wavefront or antenna pattern.
  - Same process can be used to measure changes in performance of the elements and compensate
  - Really need to model forward in time (measure at time $t$, use at time $t+\Delta t$)
How the ACE array works

Incoming signals arrive staggered in time and phase, depending on direction of arrival.

Element radios adjust time delay, phase, and amplitude to compensate for:
- Direction of arrival
- Physical position
- Radio performance variations

Adjusted element signals are transmitted by radio to a common receiving point where they are summed.

Metrology on each element measures position of element and RF performance

Wireless “LAN” used for control, status, and software loads
But, there’s a few challenges

- Historically, phased arrays have used prior calibration and precomputed adjustments
  - A central control processor turns “point the beam that-a-way” into thousands of individual commands to each element.
  - Some adaptive arrays have been built that add a “metrology” system that measures the element positions (or, more commonly, the position of a panel containing a number of elements)
- Huge computational problem (thousands of interacting elements, thousands of measurements, etc.)
  - Computational load such that it’s done off-line, not in real time.
- BUT… what if we put a processor and measurement equipment at each element?
- Now we have a cluster of thousands of processors to apply to the problem
  - And the internode communication is well mapped to the interelement interaction matrices!
Distributed Algorithms

The key to scalability

- Distributed Algorithms are the key to scalability
  - Computational requirements scale as # of elements
  - Computational resources added as elements added
- Interactions are generally "local"
  - Interactions Matrices are diagonally banded
  - Mutual Z, Stiffness, etc.
  - Locally calculate and update

What's my RF "state"?

What should I do?

Adjustments & Model Inputs

Where am I?
What's my orientation?
What's my RF "state"?
4 breadboard elements