Agent-Based Adaptive Computing for Ground Stations

Rodney Price, Ph.D.
Stephen Dominic

TRW Data Technologies Division

February 1998
Target application

- Mission ground stations now in development
  - Very large signal processing task:
    - uses ~10 large parallel processing machines with up to 64 processors each.
  - Very data intensive:
    - input data rates of up to tens of megabytes per second.
  - 7/24 operation:
    - time available for repairs, maintenance, software upgrades is minimal.

- Long-range plans
  - Reduce up-front costs -- do more with less:
    - the budget for today’s “big iron” may not be available.
  - Reduce maintenance costs:
    - run the ground station in “darkened” mode -- no on-site operators.
  - Increase ability to handle surges in input data rates.
Why are we doing this?

- **64 processor SGI Origin 2000**
  - 195 MHz R10000 64-bit CPU’s, 4 MB cache.
  - HIPPI networking, 800 Mb/s throughput.
- **Price: $1.1 million** (after 30% discount); estimate from SGI sales office.

- **100 processor PC farm**
  - 300 MHz Pentium II CPU’s, 2 CPU’s per machine.
  - Myrinet networking, 250 Mb/s throughput under IP, >1000 Mb/s raw.
- **Price: $360,000** (unit pricing); estimate from Dell and Myricom.

- Because the PC farm has many identical, interchangeable parts, we can build in very **adaptive**, very **fault-tolerant** behavior in the software.
Some philosophy

- Nature is full of adaptive, robust systems:

- What makes an ant colony work so well?
  - Interactions between ants are always local, never global.
  - Global behavior (bringing food home) results from the ants following simple rules in their purely local interactions with other ants.
What has this got to do with computing?

• Our computations are carried out by a system of many software agents.
  – Agents are mobile -- they can move around the network.
  – Agents always interact locally with other agents.
  – Agents are goal-oriented -- they have a task to perform.
  – Agents are aware of their immediate surroundings.
  – Agents follow simple rules to use system resources efficiently.

• Good dynamic load-balancing occurs because:
  – Each agent has a part of the overall signal processing problem.
  – The agents compete for processor cycles and network bandwidth.
  – The agents have simple heuristics for moving from processor to processor.

• Fault-tolerance and adaptive behavior are side-effects!
  – They are emergent behaviors, just as flocking is an emergent behavior of birds.
• Our current system consists of four dual Pentium Pro machines (eight processors) running Windows NT.

• We have chosen a simple but important signal processing algorithm, clustering by region-growing, as a test problem for our agents system.

• Java is our test platform.
  – On our clustering application, the Symantec Café 2.0 Java JIT compiler beats MS C++ 5.0 and is only slightly slower than Symantec C++ 7.5.
  – We use our own lightweight agents framework.
Clustering by region-growing

- The region-growing clustering algorithm proceeds in three steps:
  - Start by forming preliminary clusters of radius $d_1$ or less.
  - Remove all singleton clusters, clusters with only one point.
  - Use the centroids of the preliminary clusters as new data points for the final clustering. The final clusters have radius $d_2$ or less.
- We scatter the incoming points among the machines in the network.
Agent system architecture
Performance

- Agent overhead is about 15 - 25%.
  - Tested by running optimized sequential clustering against agent-based clustering on a single machine.
- Parallel performance: We get nearly linear speedup on up to four machines.
Adaptive behavior

- Machines can be withdrawn cleanly from a computation without compromising the results.

- New machines added to the network are discovered and put to use without operator intervention. (Other than starting an agent-host server.)
- A shutdown agent-host server “repels” all agents.
  - Clearing the shutdown makes the machine available for use.

- Agents cannot go anyplace where an agent-host server is not running.
• Unexpected thread or agent death:
  – If an agent sends another a message, and the receiving agent is gone, an exception is generated, and a new agent is created.
  – This agent may or may not finish operating on its data in time.
  – If not, some data is lost, so the result loses some accuracy.

• Unexpected processor crash:
  – Exceptions are generated on messaging, but data on the processor is lost.
  – Agents ignore the processor thereafter.

• Data processing always proceeds correctly after the next timer interval. No operator intervention is required.

• Unexpected network failure:
  – Operations continue on separate processors.
  – Partial results are generated on each machine; separate reports are generated.
Configuration management

- Each machine in the network needs an identical copy of the agent-host server.

- All application code starts execution on a single machine. The agents themselves distribute the code as the algorithm runs.
  - Then application code has to be introduced only on the machine the computation originates on.
Summary

• Our system consists of many mobile, lightweight agents.

• Fault-tolerance and adaptive behavior emerge from the competition of agents for system resources.

• Our approach addresses the requirements of next-generation mission ground stations with
  – Inexpensive COTS hardware and system software.
  – Adaptive behavior in response to load fluctuations.
  – Robust fault-tolerance, suited to unattended 7/24 operation.