Evaluating Performance Of Distributed Computing Technologies – HLA and TSpaces

Ratan Guha, Joohan Lee, Oleg Kachirski
School of Computer Science
University of Central Florida
Acknowledgements:
ARO Grants: DAAA19-01-1-0502
& W911NF04110100
NSF Grant: EIA 0086251

Outline of the Presentation

- Cluster Computer in Parallel & Distributed Simulation
- High Level Architecture and T Spaces
- Intrusion Detection Systems
- BPNN Parallel Neural Network Classifier
- Implementation Using HLA and TSpaces
- Performance Comparison
- Concluding Remarks

Cluster Computers for PADS

- Issue: Distributing the execution of a discrete event simulation program over multiple processors
- Preferred architecture: Parallel computers
  - Low communication latency
  - Shared-memory model-capable
  - Expensive
- Cost-effective architecture: Cluster computers
  - Beowulf Cluster
  - Increased communication latency
  - Best price/performance ratio

High Level Architecture

- IEEE Standardized framework for DoD simulations
- Modular, component-based software architecture
- Incorporates subsystems for communication and data sharing, synchronization, and time management
- Provides standard integration architecture for separate and remote applications
- Facilitates reuse of simulation components
- Multicast network protocol for reliable data exchange

T Spaces (IBM)

- Tuple Spaces - Older distributed communication paradigm
- Best suited for shared-memory parallel computers
- Consists of a set of network communication buffers (tuple spaces) and a set of APIs for accessing them
- IBM developed Java-based TSpaces based on Tuple Spaces
- A viable choice for a heterogeneous distributed system
- Simpler user-side coding than with HLA

Salient Features of T Spaces

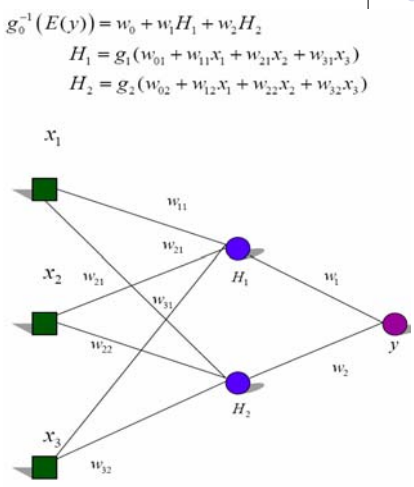
- Tuplespace operator superset
- Dynamically modifiable behavior – new operator to be defined dynamically
- Persistent data repository – T Spaces operations are performed in a transactional context ensuring the integrity of data
- Database indexing and query capability
- Access controls – users can establish security policies
- Event notification

Intrusion Detection Systems

- Anomaly and misuse detection models
  - Analyze audit data and network packets
  - Establish normal operational profile and detect deviations
- Alternative approaches:
  - Rule-based intrusion detection (Snort)
    - Maintains set of Rules of what is considered intrusion
    - Maintains a database of known attacks
  - Neural network techniques
    - Requires lengthy initial training period
    - Builds normal operational profile

Back Propagation Neural Network

- Predict the value of y for a given input (x_1,x_2,…,x_n)
- Known (x_1,x_2,…,x_n,y) records are used to train the network and find optimized weights
- Normally more than one iterations of training



Parallel BPNN Classifier

- Data mining within large datasets places high demand on computational resources
- We model BPNN - a back-propagation neural network training algorithm for an intrusion detection system using HLA and TSpaces
- Task of BPNN is to build a predictive model (classifier) of network data
- Training data provided by Lincoln Labs (Lippmann 2001) – industry standard intrusion dataset

BPNN Classifier Training

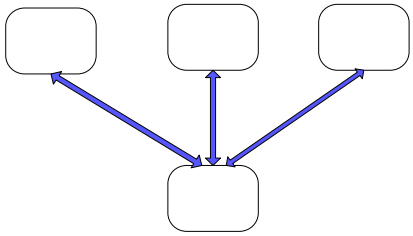
- A finite dataset of 2 million records is partitioned equally among N processors
- Each processor is an independent self-contained computer (node)
- Master node - responsible for task allocation and result unification
- Training method for a cluster computer is based upon set partitioning and epoch-based weights update schemes
- Benefit - reduced amount of inter-processor communication needed to distribute data

BPNN Training (cont-d)

- During every epoch, master process creates a series of task requests for worker processes
- Each request contains information on a fraction of the entire dataset to be processed
- Worker processes process fractions of the dataset and communicate results back to the master
- Master process then aggregates all partial results and evaluates quality of training
- Process repeats until desired quality is reached

Parallel Back Propagation Neural Network

- Divide training data set into smaller sets
- Each process trains the network use a smaller set
- The master process collects and redistributes the new weights after each iteration of training, so the communication is between the master process and other worker processes



HLA/RTI Implementation

- 16-processor cluster computer architecture
- BPNN Federation created
- Worker and Master federates communicate via multicast connection
- Partitioned subsets are allocated to specific worker processes and distributed directly by master
- Implementation assumption: homogeneous nodes with equal configuration and resources available
- Has to be modified if moved to a heterogeneous distributed environment

TSpaces Implementation

- Same 16-node cluster computer
- Master doesn't communicate directly with Worker processes
- Jobs are placed in a queue and picked up by first available Worker process
- Concurrent access is controlled by TSpaces API
- Simpler implementation logic
- Can easily be ported to a heterogeneous distributed environment – accounts for variations in each node's resource availability and usage

Hardware Architecture

(A) Ariel Cluster (SUN):

- SUN Solaris OS
- 32 dual-processor P4 2.6GHz Nodes
- 1 GB Memory on Each Node
- Fast Ethernet Switch

(S) Scerola Cluster (UCF):

- Linux RH9 OS
- 64 AMD T-Bird 900MHz Nodes
- 1 GB Memory on Each Node
- Fast Ethernet Switch

http://www.cs.ucf.edu/~jlee/Lab/facility.html

HLA Algorithm – Worker Class

```
Class Worker {

    Connect to RTI;
    Create / Join Federation Execution;
    Publish / Subscribe to Interaction Classes;
    Retrieve Training Data;

    Loop {
        Receive Work Order;
        De-serialize Parameters;
        Build / Verify Work Order Set;
        Compute BPNN Weights;
        Serialize Parameters;
        Send Result to Master;
    }
}
```

HLA Algorithm – Master Class

```
Class Master {
 Connect to RTI;
 Create / Join Federation Execution;
     Publish / Subscribe to Interaction Classes;
     Read Initial Weights;
     Get Start Time;

  Loop {
            Compute New BPNN Weights {
                      Send Out Work Orders to Workers;
                      Receive All Results;
            }
            Test Result  {
                      If Result Successful, Display execution time and Quit;
                      Else Continue;
            }
    }
}
```

TSpaces Algorithm

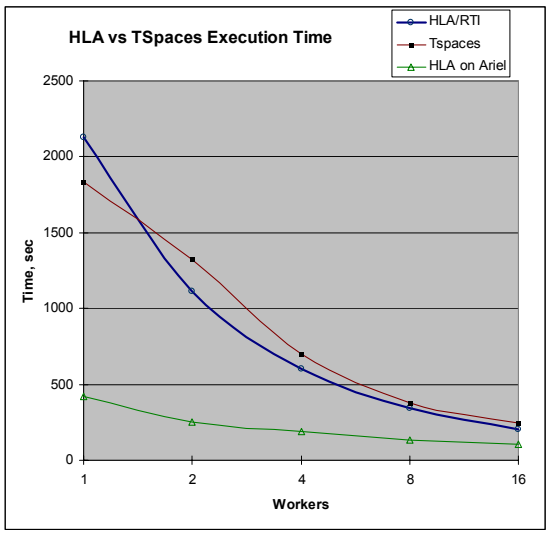
- Similar to HLA with some differences:
- Master doesn't communicate with a specific Worker process, but places a work order in a virtual shared memory
- Master then notifies all clients that data is available for pickup (via signaling or polling)
- There is no addressing scheme and no way to specify a particular processor to send data to
- Tuple space clients continuously scan for data that matches specific template

Parallel BPNN Performance

Nodes/Exec Time (sec)	HLA/RTI (S)	TSpaces (S)	HLA/RTI (A)
1	2129.37	1837.13	417.83
2	1115.98	1323.58	251.16
4	601.24	697.64	188.95
8	345.02	380.83	134.93
16	204.23	243.49	105.32

Execution times to complete neural network training for a dataset of 2,000,000 records

Parallel BPNN Performance



Hardware Configuration:

We have used:

(S) = Scerola Cluster (16 x 900MHz Athlon, 1GB RAM)

(A) = Ariel Cluster (16 x dual P4 2.6GHz, 1GB RAM)

Results Analysis 1

- HLA shows consistent improvement in run-time over TSpaces implementation
- Implementation complexity using HLA is higher
- For a one worker case, HLA is slower than TSpaces
  - Overhead of HLA as a more heavyweight solution
  - Internal optimizations of TSpaces API when system run on only one processor
- Direct communication is, as predicted, much faster than a shared-memory emulation on a cluster

Results Analysis 2

- When moving to distributed system implementation, overall cost of system usage increases
- Using HLA/RTI with one processor yields an execution time of 2129 processor-seconds
- Total system busy time using 16 processors yields an execution time of 3267 processor-seconds
- If processor time is an expensive resource, a careful cost analysis and planning has to be performed prior to implementing the distributed system

Results Analysis 3

- Dual-processor cluster computer configuration:
  - Java Virtual Machine efficiently utilizes dual-processor configuration
  - Execution of computationally-intensive tasks is much faster without further modeling by the researcher
  - Cheaper alternative to upgrading to 2 independent single-processor nodes
  - Results demonstrate cost-effectiveness of using a dual-processor cluster configuration
  - Requires careful planning of the system, as processing time may no longer be a bottleneck

Summary

- Compared implementations of a data mining problem related to intrusion detection using two distributed technologies – HLA and TSpaces
- HLA is a more heavyweight solution, requiring more effort in implementation and more overhead but offering better performance for a large system
- TSpaces is smaller, simpler to use and offers intrinsic load balancing for heterogeneous systems, at the expense of performance
- Careful analysis of system capabilities and associated costs should be performed when designing a distributed system and selecting most appropriate distributed technology