Parallel Agent Based Simulation on PC Cluster

Seif Haridi
Konstantin Popov
Mahmoud Rafea
Fredrik Holmgren
Swedish Institute of Computer Science
www.sics.se/~seif
seif@sics.se
Abstract Architecture

Scenario
- Agents
  - Behavior
  - State
- Landscape
  - Placeholder
  - Physical Space

Simulation System
- Simulation Manager
  - Interaction
  - Scheduling
- Workers
  - Behavior invocation
- Monitors
  - Track behavior
Example simulation model

- A bottom up agent based simulation
- Agents: users, sites representing web surfers, and web sites
- Landscapes: users are connected in Small World network (social network, 1-lattice), sites are connected in a similar network
- The model of time is discrete
- The implementation system is Mozart, language is Oz (http://www.mozart-oz.org)
Oz and Mozart at a glance
http://www.mozart-oz.org

• Oz Language
  - Multiparadigm language, strong support for compositionality and concurrency
  - Component-based programming
  - Simple formal semantics and efficient implementation

• Strengths
  - **Concurrency**: ultra lightweight threads, dataflow
  - **Distribution**: network transparent, network aware, open, fault detection
  - **Inferencing**: constraint, logic, and symbolic programming
  - **Flexibility**: dynamic, no limits, first-class compiler
Web Word of Mouth Model

User characteristics:

• Have preferences for specific categories of content
• Participate in “local” social networks
• Maintain a portfolio of frequently visited sites
• Have memory about visited sites and their perceived utility
• Have an evaluation method in order to evaluate sites they visit.

User behavior

• Asks friends to propose their favorite sites and visit them.
• Visits some sites from his portfolio.
• Surf along the links of the already visited sites.
• Replace a site in the portfolio by a new site if that new site maintained a higher utility for a longer period of time.
LEFT: Distribution of web sites by size

RIGHT: Progressive evolution of the histogram “number of sites/number of users” towards a power-law distribution
Abstract Architecture

**Scenario**

Agents
  - Behavior
  - State

Landscape
  - Placeholder
  - Physical Space

**Simulation System**

Simulation Manager
  - Interaction
  - Scheduling

Workers
  - Behavior invocation

Monitors
  - Track behavior
Component-based Programming

• Implements abstract data-types
  - Encapsulation of internal state
  - Interface of external operations
  - Instantiation (component instances, modules)

• Compositionality
  - Specifies required (imported) components in terms of their operations only
  - Decouples linking of the exact modules from import specification.
  - Connecting component instances are done by the component-manager at runtime (allows linking different component instances depending on runtime conditions)
Component-based Programming

Component instance

Interface

External world
Sequential Simulator Components

- Landscape
  - Small World (SM) graph
  - Internet Sites (IS) graph
  - Other graphs
- Agent state collections (set of agents)
  - Users collections, and Sites collections
- Agents behaviors
  - ExploreSites, GetRecommendations, UpdatePortfolio, etc
Architecture Sequential - Simplified (component instances)

Simulator Control → (One-step) Behavior Simulator

A → B

A uses Ops of B

Sites → Users
Users

- Places the User agents on the SW graph
- Provides the Ops. of the Users abstract data-type needed to express behaviors
- Sites done in similar way
One-step behavior simulator
- Iterates over agents (users) behaviors as specified by CB

Composed Behaviors CB

Beh 1
Beh n
Sites
Users
Parallel Simulator Requirements

- **Goals**
  - Goal: 1000000+ sites/users, 1000+steps AFAP
  - Today's sequential system takes ≈1min for 10000 sites/users +100 steps on a 1Ghz processor
  - WORSE: *at best* linear memory requirements: ≈ 0.25Gb per 100000 sites/users
  - Developing techniques & tools for high performance parallel computation in Mozart
  - Study and improvement of [distributed] Mozart
Parallel Simulator

- Sites and Users collections are partitioned among N computers (N workers)
- A Manager computer is responsible for creating Sites and Users component instances and partitioning them to the workers according to their relative performance
- The behavior-simulator component is unchanged
- The User and Sites component instances are wrapped using a distribution abstraction that allows transparent access to remote user-agents and site-agents
Architecture Parallel Simulator

Manager

Simulator Control

One-step Simulator

Sites#1

Users#1

One-step Simulator

Sites#n

Users#n
Distributed Users

- Abstracts the network
- Services requests from remote workers
- Send requests from local worker to remote ones

![Diagram of Distributed Users]

- Remote users proxy
- Local users server
- Local Users
  - User Collection State
- Small World Graph

Network
Simple set-up

- Manager partitions user and site agents according to the workers performance
- Manager initiates work at each time step
- Each worker performs the work as specified by its one-step simulator
- Workers report to manager at the end of each time step
- This process is reiterated
Ways of performing Ops. on agents (sites/users)

- Remote (operation to data)
  - A request to perform an operation on a site/user is sent to the responsible worker

- Replication (data to operation)
  - State of agent is replicated to requesting site
  - Works for stateless (immutable) data
  - Eager / Lazy

- One-Step Caching (data to operation)
  - State is cached at the worker when requested
  - State updates are done locally
  - At the end of a simulation step operations are merged/performed at the agent’s worker
  - The cache is evicted (cleaned)
Sequential, 10000 sites
users/time

![Chart showing time in seconds for different user counts.](chart.png)
Threading the Behavior Simulator (BS)

- Sequential BS is running a single thread.
- In the Parallel Simulator, multiple threads are executed to hide network latency.
- This can be done in Mozart without changing 'Behaviors' due to Mozart's dataflow property.
- A thread issues operations sequentially, blocks transparently on variables until bound.
- How many threads per worker? depends on network latency.
We Want to study the scalability of the example application w.r.t. Problem size.

As can been seen on next slide 1000K agents/ 16 workers experiment take similar time as 62,5K agents/one worker
Speed-up

- The speed-ups are good.
- The most general case with one-step caching obtains 10 times speedup on 16 workers.
- The speedup increases with larger problem size.
Dataflow synchronization

• A worker at step N can serve requests for steps N-1, and N
• Serving a request at step N+1 state is delayed until the worker advances to N+1
• For any worker at step N, each other worker is either in step N-1, N, or N+1
Dataflow synchronization

- At the end of step N-1, a worker sends a sync(N) message to all other workers.
- The worker waits for all sync(N-1) messages from all other workers, before starting step N.
- For any worker at step N, each other worker is either in step N-1, N, or N+1.
Barrier vs. Dataflow Synchronization

- 10k sites, 1M users, caching, 16 workers
  - 426 sec. with dataflow synchronization
  - 500 sec. with barrier synchronization
Synchronous Garbage Collection

- 10k sites, 1M users, caching, 16 workers
  - 426 sec. with synchronous GC
  - 491 sec. with asynchronous GC
Conclusions

• Component-based programming is essential for flexible agent-based bottom-up simulation.
• It is possible to simulate large number of agents 1000k agents using the right techniques on cheap PC clusters.
• Mozart’s network transparency, dataflow synchronization, light-weight threads and component-based techniques eases the application development.