Christmas Lecture

Large Scale Applications of
Concurrent Objects
-Second Life, Twitter Systems, and Molecular Dynamics Computing-

Akinori Yonezawa
What is this talk all about?

• ... talking about a way to construct a large software system on (massively) parallel distributed computers
A Software System is...

• A software system is composed of many software modules (parts).
  – Just like a car is composed of many physical parts.

• Modules (parts) must be:
  – fairly independent from its surrounding components,
  – composable to form a larger module using other modules
  – replacable by other module with the same functionality
  – etc.
A Software System is composed by software modules.

Web Browser System

- Communication module
- Display drawing module
- Module receiving user input
Since 1950’s, the following forms:
- subroutines
- functions
- procedures
- abstract data types
- etc

Since the advent of Java language, the “object” form of software module is prevailing in software system construction !!!
What’s an Object

- Data/data-structures and methods using the data are encapsulated together in an **Object**!!
- An object encapsulates data and methods using the data.
Concurrent Object is a Generalization of Objects!!

I came up with this idea in mid 70’s.
Concurrent Object (CO)

1. CO encapsulates a statefull object + a single cpu/thread.

2. COs transmit/exchange messages concurrently and asynchronously.
Both Modeling and Programming

In mid 70’s, I started being interested in:

1. modeling worlds and simulate them on computers!

2. powerful programming frameworks!
in computer science comm. Of Mid 70’s,

• IC is maturing and time-sharing systems are in use
• ARPA Net is expanding...
• A bright future of VLSI was envisaged.

• Time to prepare ourselves for abundance of inexpensive microprocessors
• Start developing technology for utilize them
My view in 70’s:
Modeling and Simulation
by Concurrent Objects

Real World

things, people, computers &
their interactions

WEB/net

Concurrent Objects &
their message passing

modeling
representing
Crucial Features of Concurrent Objects

1. Message/event-driven activation/processing
2. Asynchronous message transmission
3. Information sharing only through message passing
4. Control flow and message flow coincide
5. Massive parallelism through massive concurrent objects
6. Enables safe use of safe light-weight processes
   - No need for lock/unlock operations
Parallelism in a domain/world can be naturally/directly modeled with COs

In domains,

- Rain drops fall in parallel.
- Things exit in parallel.
- Parallelism by an ensemble of many COs
- State change of a thing is reflected by state change of the CO.
  - COs move in V-space as things moves in space.
- COs interact with message transmissions as things interact.
1. Message/Event Driven

Concurrent Object

M

Concurrent Object

thread
2. Asynchronous message passing

Type 1

Type 2

Type 3 (future)
3. Information sharing is possible, only through message passing,
What can be modeled and represented by COs

- Mass, particle structures
- Human, agent,

More abstractly...

- Virtual processor (VP), light-weight thread
- Fine parts and processing units obtained by partition of large problem
Large Scale Applications of COs

1. Second Life system
2. Twitter system
3. NAMD, a nano-bio molecular dynamics simulator for supercomputers
Which CO’s advantages enable large scale applications?

1. Natural modeling power
   - Second Life -- simple and powerful description

2. Parallel programming with safe high performance
   - Twitter, Facebook

3. Parallel programming with high performance and high productivity
   - Charm++ → NAMD → analysis of molecular structures of Swine Flu
Second Life
Back to Original Motivation of COs

Concurrent Objects & Message Passing

Modeling & Representing

Real World

WEB

Entities, people, machines & their interactions

Linden’s Second Life ...

is a natural outcome from the motivation of Cos.
Concurrent Objects in Second Life

- Linden’s online Virtual World that millions of people participate in!

When a man gets in Ferris Wheel, it starts to rotate.

- Objects and avatars are represented and programmed as concurrent objects!!

Image from “Programming Second Life with the Linden Scripting Language” by Jeff Heaton (http://www.devx.com/opensource/Article/33905)
COs in Second Life


- Objects and avatars cooperate and coordinate each other by exchanging messages.
- Each object or avatar is programmed to
  - Have its own state,
  - Have its own method to respond to an incoming message,
  - Have different responses to different states, and
  - Have its own thread.
- About 2 millions of objects are programmed in Second Life and they are in action.
Second Life Teleporting Demo
Twitter
Joichi Ito (伊藤 穣一) talks about Twitter
Twitter System

Over 100 millions people are using it
COs in Twitter

Implementing Twitter System requires:
a huge number of twitters need to be processed in parallel.

accomplished by using a huge number of COs!!

The CO/actor mechanism in Scala language is employed!

Ruby on Rails was used in an old version of Twitter.
Implementation of Twitter System

- Requests from clients are processed by COs
- A large number of twitters are processed by a large number of COs
- COs enable a simple construction of the stable and fast system.

Each processing code is CO
Why COs are Used?

Robey Pointer says:
When a client logins, a CO is created and waits for twitters to come.

• Use of COs makes the program very simple!!
  — owing to the nature of message-driven and asynchronous messages
  — Krestel (Core of the system) is written by 2000 lines of code

• Enables fast processing, and load balancing
  (At peak, 10 thousands twitters are processed)
  — a huge number of COs make load balancing possible

• No more than one thread is contained in a single CO.
  — Lock/unlock operations are unnecessary!!

• Handling the twitter queue is possible by a large number of threads for an ordinary load, but still difficult and unsafe for lock/unlock operations
  --- Program safety is hard to guarantee!!
KESTREL
(http://github.com/robeyy/kestrel)

• Core software modules in Twitter System which handles client messages(twitters)

• Used as backend for Twitter System
  http://blog.twitter.com/2009/01/building-on-open-source.html
Kestrel

• Message queue hand system written in language Scala
  
  – CO/actor facilities supported in Scala are extensively used.
  – The code is only 2000 lines long!!
  – Tens of thousands of parallel processing by CO/actors
Scheme of Kestrel

• Requests from clients are processed by CO/actors
  – For each client, a CO/actor is dynamically created.
  – E.g., 7000 CO/actors for 7000 login clients in parallel
NAMD
(Nano-scale Molecular Dynamics Simulator)
NAMD[1996]

- Nano scale molecular dynamics simulator for supercomputers
- Received Gordon Bell Prize in 2006
- Developed jointly by two groups in Illinois Univ. Prof. Sanjay Kale (Computer Science) Prof. Schulten (Theoretical Biology)
- Written in Concurrent Object-based system Charm++
  “NAMD is a parallel, object-oriented molecular dynamics code designed for high-performance simulation of large biomolecular systems.”
- Simulate step by step, each step representing 1 fs.
- At each step, calculate forces exerted on all molecules and update their positions and velocities
This still from a Quicktime movie represents a view of the drug buried in the binding pocket of the A/H1N1 neuraminidase protein.
Charm++[1993]

- CO-based language system for parallel/supercomputers developed by Prof. Kale.
  -- many references to our CO-based language ABCL
- Used for development of NAMD, OpenAtom (quantum chemistry), ChaNCA (Galaxy generation)
- Operational for machines with 1000~10000 CPUs
- Has a framework for dynamic load balancing by moving Cos from node to node
- Applications written in Charm++ are
  - Utilizing and consuming 15%~20% of computing resources into two major supercomputing centers in US
Goals of Charm++

CO-based programming system for supercomputers
http://charm.cs.uiuc.edu/

• Goals:

1. Improve performance of parallel applications for regular and dynamic problems

2. Improve productivity of programmers and programming
Analysis of Molecular Structure of Swine Flu by NAMD

Showed:

- Structural changes of A/H1N1 protein induce anti-tamiflu effects.
Towards Many-Core Machine Era
Multi-Core (Many-Core) Chip

• 4 ~ a few hundreds of CPUs on a single chip

• Laptop with 30~40 core machine, in 5 years
New Type of Supercomputers

• 100K ~ 1000K machines (node) connected each other!!!
• Each node contains 16> cores
How to Program
Many-Core Machines
and Supercomputers

• Programming easily and efficiently is a big problem!!

• Main problems:
  – Synchronization between processes (CPUs)
  – Communications between processes
  – Deadlock avoidance
  – Developing parallel algorithms
A Solution

- Concurrent-Object based Programming (CO)
- Concurrent Object-Oriented Programming (COOP)
End