A Report on the Project Darkstar
Anthropological Expedition Into the World of Massively Scaled Online Games

• Jim Waldo
• Distinguished Engineer
• Sun Microsystems Labs
A Preliminary Report from the Darkstar Anthropological Expedition Into the Unexplored Jungle of Massive Multi-Player On-Line Games
MMOs Are Different

• Different roles
  > Producers
  > Artists
  > Coders

• Different goals
  > Fun
  > Cool

• Different organizations
  > Publishers
  > Production houses
What We Look Like

Sun Microsystems

ULTRASPARC

Solaris

Java

opensolaris

STORAGETek
What They Look Like
The Numbers Are Staggering
World of Warcraft

• Approximately 10 million subscribers
  > Average subscription : $15/month
  > Average retention : two years +
  > $150 million per month/$1.80 Billion per year run rate
  > For one game (they have others)

• Unknown number of servers

• ~2,700 employees world wide

• Company is changing
  > Was a game company
  > Now a service company
Webkinz

• Approximately 5 million subscribers
  > Subscription comes with toy purchase
  > Subscription lasts one year
  > Average 100k users at any time
  > Currently only US and Canada; soon to be world wide
  > Aimed at the 8-12 demographic
    – And their mothers...

• The company is changing
  > Was a toy company
  > Becoming a game/social site company
Cultural Observations

• Games are part of the entertainment industry
  > Producers, daily rushes, story lines
  > Fun/engagement most important

• The default computing environment
  > A PC or console
  > One thread
  > One player

• Scaling and reliability have never been vital

• Low latency, not total throughput

• Not a traditional “IT” market
Riding Moore's Law

• When processors get faster
  > Games play faster
  > Things can be more complex

• When GPUs get faster
  > Better visuals
  > More engaging play

• Games machines are supercomputers
On-Line Games Change Everything

• Scale and reliability needed
  > One call to customer service = ~3 months subscription
  > Slow games are not fun
  > Server crashes impact multiple players

• Capacity management is hard
  > Hit games need to scale up
    – Sometimes faster than human time
  > Duds need to scale down

• Chip architectures are changing
  > Threads, not clocks
Current Scaling Techniques

• Geographic Decomposition
  > One server = some geographic area
    – Island
    – Room
  > Need to decide scale during production
  > Get it wrong, game play impacted

• Shards
  > Copies of the game world
  > Allow multiple people to do the same thing
  > No communication between shards
    – Bad for guilds
Reliability Techniques

• Snapshots
  > On occasion, dump state to a database
    – Dumping state sucks cycles
    – Done during transitions (run the video...)
    – May not happen frequently

• Otherwise, state kept in memory
  > Faster access
  > Lost if the server is lost

• Considerable game play can be lost
The New Environment

• Multi-core machines
  > Clocks aren't getting faster
  > Cores are multiplying
  > Only works for highly concurrent programs

• Highly distributed
  > Need servers to work together
  > Need to be able to dynamically scale
Project Darkstar Goals

- Support Server Scale
  - Games are embarrassingly parallel
  - Multiple threads
  - Multiple machines

- Simple Programming Model
  - Multi-threaded, distributed programming is hard
  - Single thread
  - Single machine

- In the general case, this is impossible
The Special Case

• Event-driven Programs
  > Client communication generates a task
  > Tasks are independent

• Tasks must
  > Be short-lived
  > Access data through Darkstar

• Communication is through
  > Client sessions (client to server)
  > Channels (publish/subscribe client/server-to-client)
Game Architectures

- Powerful clients
  - Lots of graphics
  - Lots of state
- Simple servers
  - Abstract model of the world
  - Does as little as possible
- Communication protocol
  - Small messages
  - Best guess then repair
  - No peer-to-peer
Project Darkstar Architecture
Stack Architecture

- Task Service
- Client Session Service
- Channel Service
- Other Services
- Data Service
Dealing with Concurrency

• All tasks are transactional
  > Either everything is done, or nothing is
  > Commit or abort determined by data access and contention

• Data access
  > Data store detects conflicts, changes
  > If two tasks conflict
    – One will abort and be re-scheduled
    – One will complete

• Transactional communication
  > Actual communication only happens on commit
Project Darkstar Data Store

- Not a full (relational) database
  > No SQL
  > Assumes approximately 50% read/50% write
- Keeps all game state
  > Stores everything persisting longer than a single task
  > Shared by all copies of the stack
- No explicit locking protocols
  > Detects changes automatically
  > Programmer can provide hints for optimizations
Project Darkstar Communication

• Listeners hear client communication
  > Simple client protocol
  > Listeners established on connection

• Client-to-client through the server
  > Allows server to listen if needed
  > Very fast data path

• Mediation virtualizes end points
  > Indirection abstracts actual channels
Dealing with Distribution

- Darkstar tasks can run anywhere
  - Data comes from the data store
  - Communications is mediated
  - Where a task runs doesn't matter
- Tasks can be allocated on different machines
  - Players on different machines can interact
  - The programmer doesn't need to chose
- Tasks can be moved
  - Meta-services can track loads and move tasks
  - New stacks can be added at runtime
The End Result

• Simple and familiar programming model
  > A single thread
  > A single machine

• Multiple threads
  > Task scheduling part of the infrastructure
  > Concurrency control through the data store, transactions

• Multiple machines
  > Darkstar manages data and communication references
  > Computation can occur on any machine
  > Machines can be added (or subtracted) at any time
Current Status

• Multi-node version available
  > Open source (GPLv2)
  > Commercial license under development

• Working on performance, reliability
  > Caching data
  > Failure recovery
  > Add/delete nodes
Current Questions

• Characterizing workload
  > Games are secretive
  > Makes it hard to know performance

• Data access break-even point
  > Memory access is always faster
  > Data store allows multiple machines
  > When do we get the same/better performance
    – Maybe never
How Much Can Be Hidden

• Lots
  > No explicit locking
  > No need to identify critical sections
  > Looks like single-threaded code

• But not all
  > Data design for concurrency
Is It Computer Science?

• Important questions around
  > Concurrent programming
  > Reliable systems
  > Dynamic distributed systems

• Not the answer, but an answer

• And it is fun...
A Report on the Project Darkstar Anthropological Expedition Into the World of Massively Scaled Online Games

• Jim Waldo
• jim.waldo@sun.com