PROJECT DARKSTAR

Concurrency, Distribution, and Server Scaling

Jim Waldo
Distinguished Engineer
Sun Microsystems Labs

Chips are Changing

- Clocks aren't getting faster
  - Design complexity constraints
  - Energy efficiency worries
  - Thermal problems
- More cores, more threads
  - Cuts design effort and cost
  - Saves energy
  - More heat efficient
- Not faster, but more
  - Any task takes longer
  - Multiple tasks run in parallel

Games are Changing

- From single player to multi-player
  - More network traffic
  - More opportunities to cheat
  - More player interactions
- It's not just the game anymore
  - Social interactions keep players around
  - Coordinating guilds adds complexity
- Isolation is not possible
  - Players want to interact
  - Loads, failures make them interact unintentionally

New Requirements

- Scale
  - Millions of players
  - Bursts of load
  - Need to balance
- Sharing
  - Players want to interact
  - Interaction patterns are unpredictable
- Reliability and security
  - Players want to play, not call customer service
  - You are a game company, not a service provider

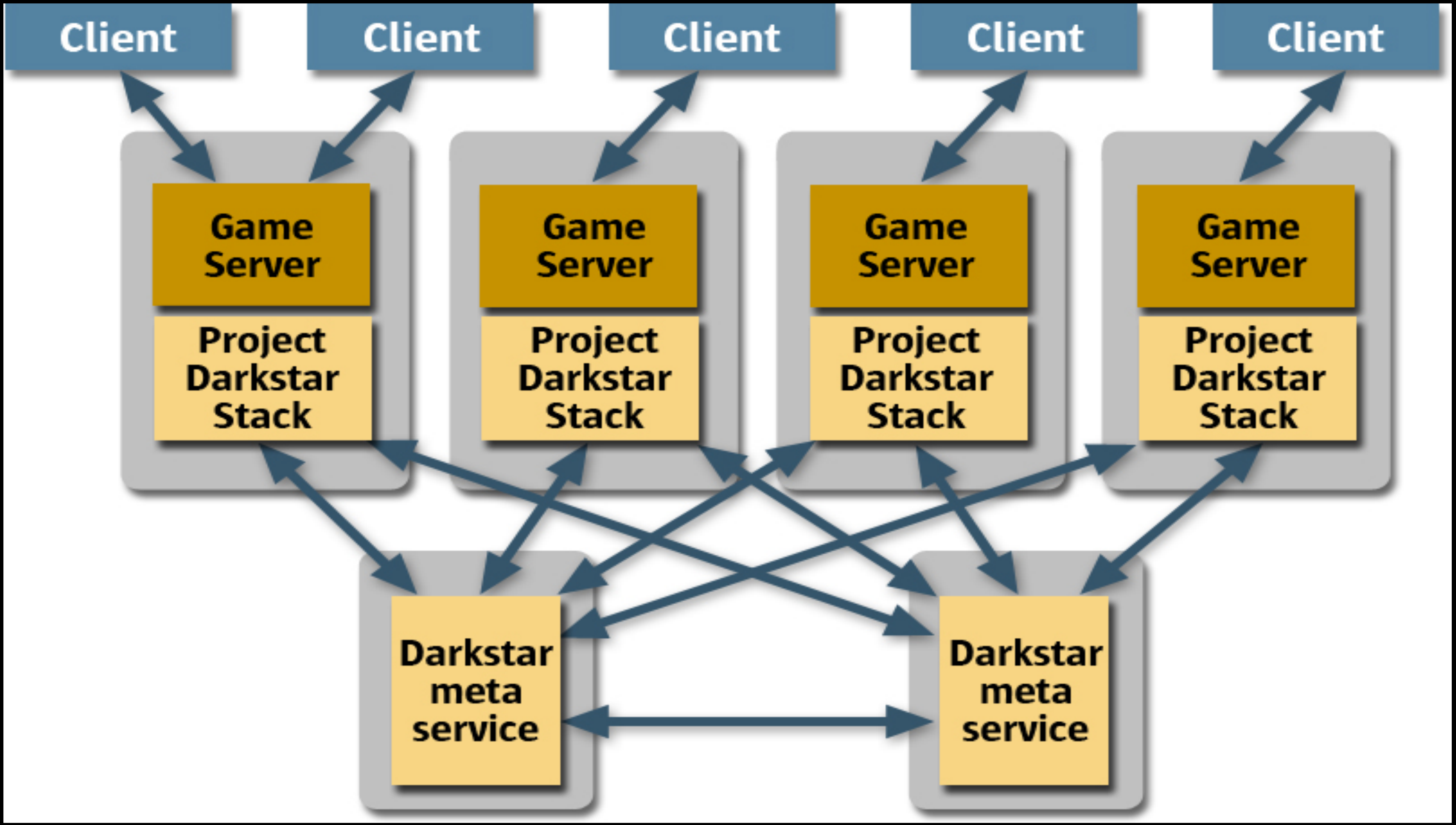
Project Darkstar Goals

- 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)

Project Darkstar Architecture



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 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

- Single node version available
  - http://ProjectDarkstar.com
  - GPLv2 license
  - If you don't like GPL, talk to us...
- Supports
  - Multiple threads
  - Transactional data storage
  - Same API as multi-node (mostly)
- Multiple machines
  - Working in our lab
  - Currently measuring, optimizing
  - Should be available soon

Why Sun?

- We make multi-core machines
  - We don't know how to program them either
  - This is a way to find out
- We do distributed systems
  - Lots of experience in the enterprise case
  - Games have a different set of requirements
- Darkstar is a lab project
  - High risk, high reward
  - Research ideas, product quality software
  - Any company large enough to fund a research lab...

Why Games?

- Games are
  - Embarrassingly parallel (we don't embarrass easily)
  - Unencumbered by legacy code
- We avoid
  - Expectations from current customer base
  - Corporate antibodies
- Fun
  - Willing to take risks
  - A different programming culture
  - Who would you rather hang out with?

Why GDC?

- We can't do this alone
  - We don't have game expertise
  - We don't ship games
- You can't do it alone
  - Distributed computing is hard
  - Concurrent computing is harder
- We can do it together
  - A Darkstar developer community
  - We learn from each other
  - We each do what we can do well
  - We let the others do what we can't do so well

The Darkstar Community

- Network Community
  - Articles
  - Forums
  - Projects
- More to do than we can do
  - Tools
  - Integration
- Community and open source
  - Core is not (currently) community development
  - Other projects are

How You Can Join

- Visit the website
  - http://ProjectDarkstar.com
  - Join the discussion
  - Tell us what we are doing wrong
  - Tell us what we aren't doing
    - Maybe do it yourself...
- Find us here
  - If the badge says “Sun”, it’s a Darkstar person
  - Join us tonight (Thirsty Bear, 7 p.m.)
    - The first drink is on us

How You Can Contribute

- Contribute to the community
  - Try the core code
  - Scratch your own itches
  - Contribute ideas
    - What else is needed?
    - What isn’t done right?
  - Contribute code
    - Tools
    - Utilities
    - Benchmarks
- Everyone wins
  - Software is better

PROJECT DARKSTAR

Concurrency, Distribution and Server Scale

Jim Waldo
jim.waldo@sun.com