PROJECT DARKSTAR

Project Darkstar Architecture

Jim Waldo
Distinguished Engineer
Sun Microsystems Labs

Java
Sun Microsystems

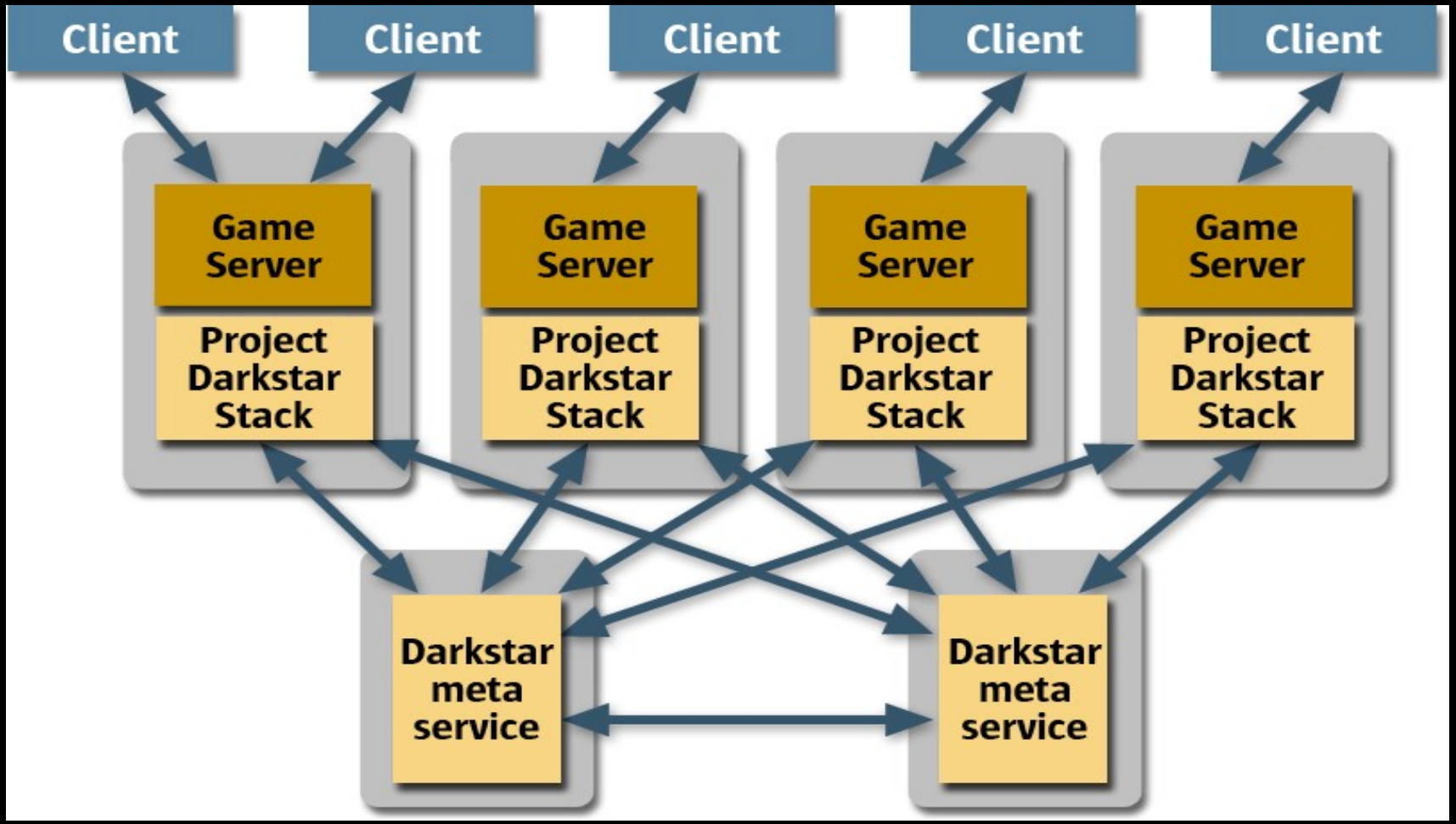
Project Darkstar Goals

- 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



Everyone and Everything Participating on the Network

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
  - Optimized for 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

PROJECT DARKSTAR

Project Darkstar Architecture

Jim Waldo
jim.waldo@sun.com

Java
Sun Microsystems