CG Consultancy

Hotel Search, Scalability, and Apache Ignite

Musaul Karim • Senior Consultant • June 2018

AGENDA

- ✓ Introduction
- ✓ Hotel Search Systems
- ✓ Architecture
- ✓ Successes & Challenges
- ✓ Questions

About Me

Software Consultant ● In-memory & Distributed Systems Specialist ● MSc Distributed Computing

Initial Career

- 2000 - Started as a C++ developer
- 2003 - Took a break to do my MSc
- 2005 - Back into world of work at Deloitte

In-Memory Systems

2007 - Fidessa

- High Transaction Order & Execution Management System
- In-house developed Distributed Cache Systems for Trade Data

2010 - Barclays

- Migrated DBMS based Risk Calculation engine to an In-Memory Cache & Compute system
- Hybrid In-house tech + Solace Systems + Oracle Coherence

2013 - Credit Suisse

- Oracle Coherence based Prime Services Risk System

CG Consultancy

- IT System Migration Projects
- Technology Assessment
- Options within the Modern Landscape
- Proof of Concepts
- Leading Follow up Development Work
- Overall Technical Architecture

Travel sector clients

- JacTravel
- OAG
- Recently started working with one of the largest travel operators

CG Consultancy

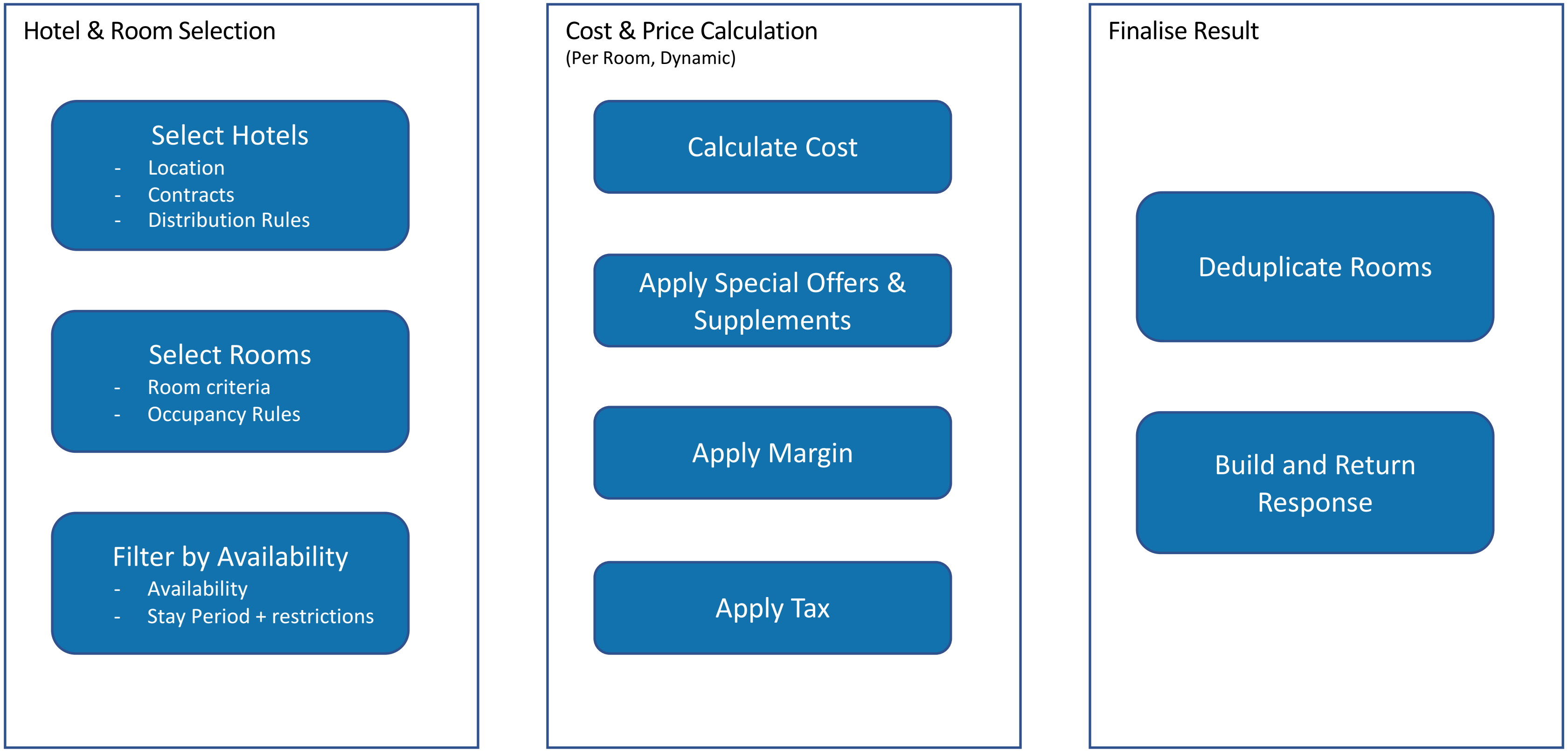
Hotel Search Systems

In-Memory Computing Summit, London • 25-26 June 2018

Hotel Search System Overview

- Handles Hotel/Room Search requests via a B2B API
- Receives updates intraday as streams as well as batches from Booking Systems and other Third Party Supplier Systems
- Returns Priced Rooms matching the Search Criteria
    - Matches Hotels based on locations searched (Can also search for specific hotels)
    - Matches Rooms based on Stay Date Availability and Occupancy requirements etc.
    - Excludes rooms based on any distribution rules
    - Calculates prices for all the room options
- Typically more I/O bound than CPU
    - It requires a large number of queries against Database Tables (or Caches) at each stage
    - Large number of calculations to be performed. i.e. they need to be done for each room / special offer / room-extras etc.

Search Journey



Hotel & Room Selection

Select Hotels
- Location
- Contracts
- Distribution Rules

Select Rooms
- Room criteria
- Occupancy Rules

Filter by Availability
- Availability
- Stay Period + restrictions

Cost & Price Calculation

(Per Room, Dynamic)

Calculate Cost

Apply Special Offers & Supplements

Apply Margin

Apply Tax

Finalise Result

Deduplicate Rooms

Build and Return Response

CG Consultancy

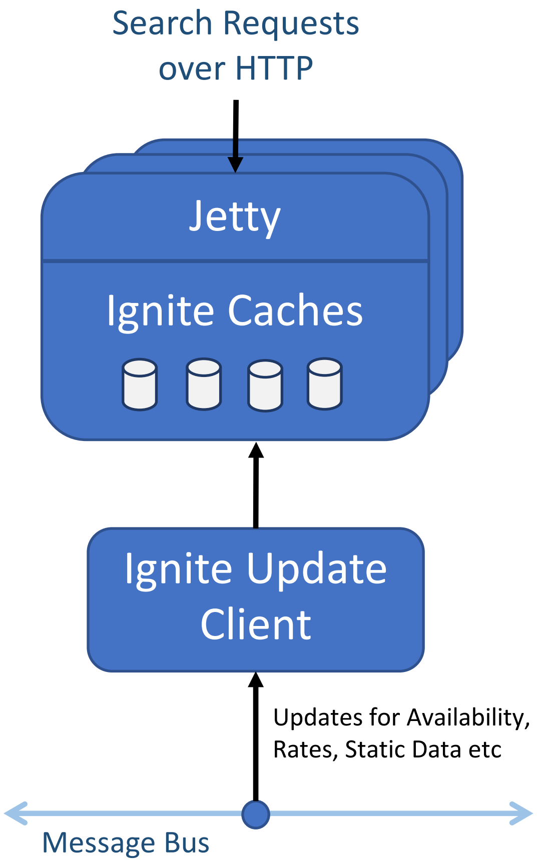
Architecture

Previous Infrastructure at JacTravel

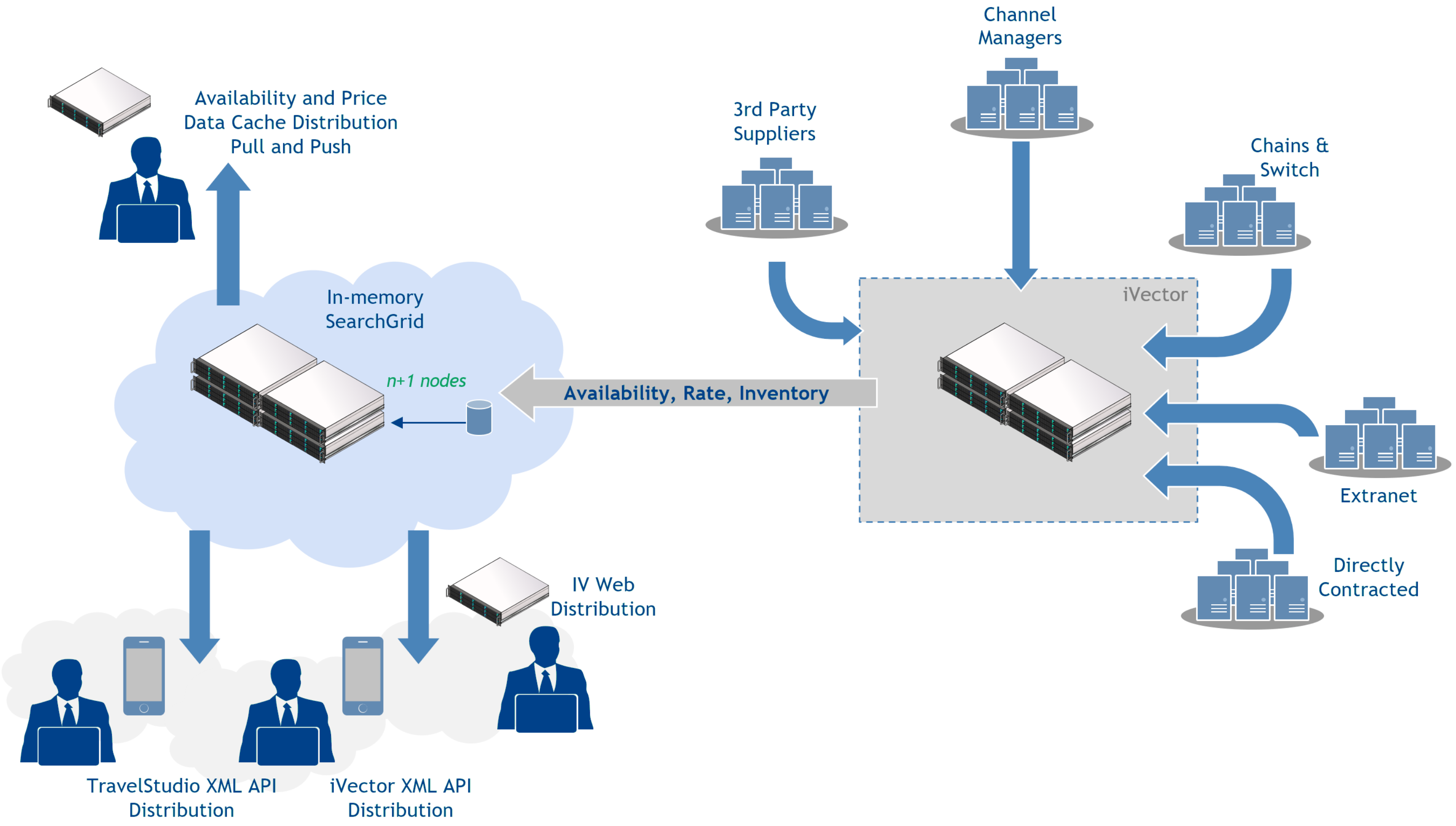
- Two Platforms
  - One retained as a booking platform (iVector)
  - The other being decommissioned (TravelSudio).
- Built on Microsoft SqlServer and IIS (VB.NET and C#)
- Over 100 SQL Server + IIS Instances
- Handled typical traffic of ~140 million searches per day
- Average Response Time of 2.5 Seconds
- Hardware upgraded as much as possible (e.g. SSDs)
- Various database optimisations considered
  - Search-specific “cache” tables
  - In-memory Tables in SQL Server.
- Infrastructure cost too high and reaching diminishing returns

New Search-Grid Overview

- Server / Cache Nodes
  - Apache Ignite embedded in Spring MVC service
  - Cluster with Fully Replicated Caches
  - Most Caches Off-Heap
  - Process consumes around 60GB memory, including a 20GB JVM heap.
  - Loaded from SQL Database
    (with no further DB at "Search-Time")
  - Requests received via Embedded Jetty and processed by an Ignite Service
  - 20 nodes handling ~300 million searches
- Update Client Nodes
  - Subscribes to a Message Queue
  - ~200k updates intraday
  - Updates Caches using a combination of Services and Ignite Data Streamers
  - Updates with no visible impact on Search Process


Search Requests over HTTP
Jetty
Ignite Caches
Ignite Update Client
Updates for Availability, Rates, Static Data etc
Message Bus

Overall Architecture



Search-Grid Internals

- ~ 50 Caches
  - Fully Replicated
  - Most are Off-Heap
- Cache Queries
  - Direct key based access where possible
  - SQL Fields and Indexes only when SQL Queries are necessary
- Search Request
  - Processed by an Ignite Service
  - SQL Fields and Indexes only when SQL Queries are necessary
  - Threads managed by Ignite Services Pool
  - Search processed using a Single thread on a Single Node
- This allows the system to be scaled up linearly

Deployment

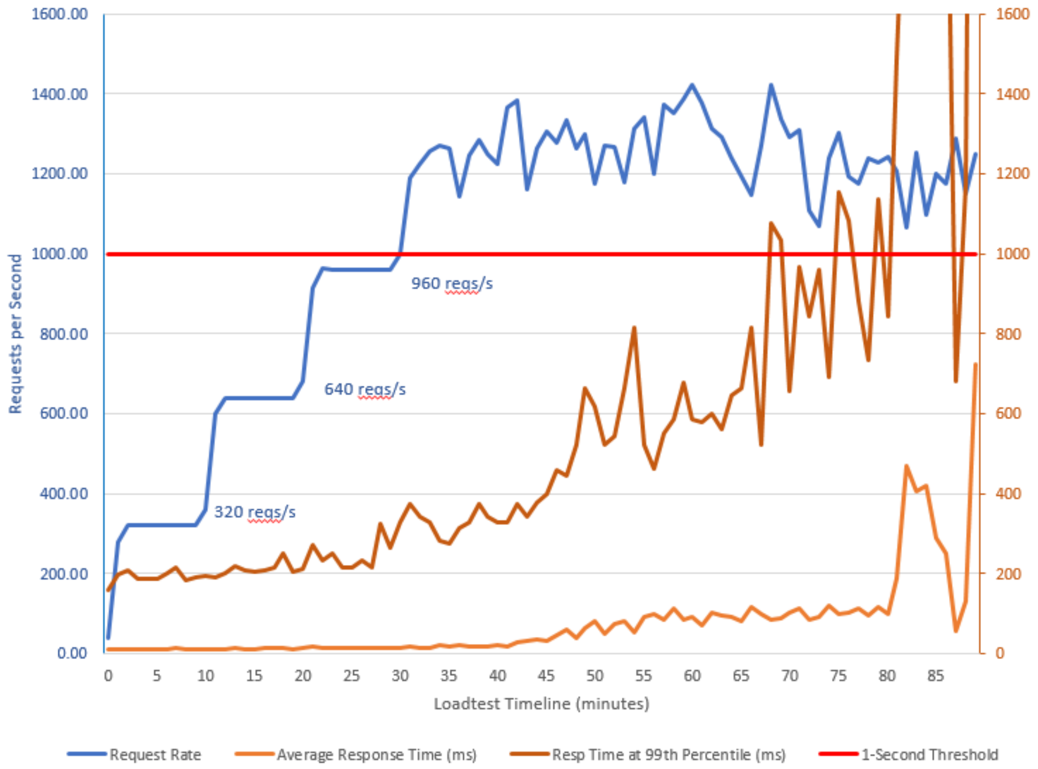
- Deployment tested on
  - Physical Hosts
  - VM / Cloud Providers: AWS, Azure, Rackspace
- Zero down-time Cluster deployment & restart
  - Starting new nodes on a separate cluster (blue/green)
  - Fully automated – orchestrated using Ansible
- Adjusting Cluster to match Traffic Volume
  - Cache Nodes can be added or removed to match Traffic Volume
  - Caches will rebalance onto new nodes
  - The Event mechanism can be used to determine when all caches are rebalanced

CG Consultancy

Successes & Challenges

Performance

- Load Test on 4 Nodes
  - AWS m4.4xlarge
  - 16 vCPU (2.3GHz XeonE5-2686)
- Request Injection
  - 8 JMeter Injector nodes
  - 320 requests/sec at each step
- Measurements Overview
  - Can sustain 960 requests / second without breaching 1-second SLA red line for 99th %
  - Average response time: ~20ms
  - 99th Percentile: ~270ms
  - Requests start queuing up beyond this rate



Migration Gains

- 90% reduction in infrastructure
- 90% reduction in Response Time
- Faster Response-Time enables new use-cases to be considered for the search process
- Linearly Scalable by adding new nodes
  - Predictability makes infrastructure / capacity planning easier
- Open Source grid-technology running on Linux
  - Aides quick and easy provisioning of ad-hoc Dev / Test environments
  - Makes it easier to have a DevOps process
- New Development Processes (BDD, TDD, CI/CD)
  - Visible correlation between user stories and code
  - Test coverage provides more confidence when making complex changes

Migration Pains

- Need for maintaining multiple systems in the interim period
  - Needs to replicate the Calculation Logic, as prices must be identical to Booking System
  - Implicit Rounding based on Database Field precision – Multiple Temp Tables
  - Existing algorithms optimised for Database Queries / Stored Procedures
- API Clients change their Search pattern/behavior after noticing the improved performance
  - Increase Search Rate
  - Increase in larger region/city searches
- Introducing new technology required new toolsets & processes for auxiliary functions
  - Replacing database based monitoring & reporting tools
  - Many options. Needed a bit of discovery process.

Supporting Services

- 3rd-party Supplier Cacheing
  - A more classical implementation of a Read-through cache
  - Reducing load on 3rd party partners
  - Smarter searches to partners based on most common search types
  - Native Persistence
- Real-Time Statistics / Analytics
  - Types of searches by clients
  - Locations being searched
  - Spikes in requests by Clients / Location
- Integration with 3rd party products for detailed analytics / visualisation

Technical Considerations

- Working with Large JVM Heaps
  - Garbage Collector Benchmarking / Comparison / Tuning
  - Development considerations to avoid long "Stop the world" pauses
- Initial Rebalancing can take a long time
  - Need to make considerations for zero-downtime deployments
- Ignite is product with a lot of active development
  - Great for getting lots of new useful features
  - Sometimes we needed help with new features, sometimes the features need some optimisations
  - When we found bugs, GridGain have helped by creating versions for us containing the fixes
- Professional support on these issues
- Developer skillset can be more business focused compared to building a platform in-house.

CG Consultancy

Questions?

musaul.karim@cgconsultancy.com

@musaul