A Scalable Master-Worker Architecture for PaaS Clouds

Vibhor Aggarwal, Shubhashis Sengupta, Vibhu Saujanya Sharma, Aravindan Santharam

Accenture Technology Labs
Bangalore, India
Outline

- Introduction
  - Master-Worker
  - PaaS Cloud
  - Motivation

- Methodology
  - Agent-based Architecture
  - Application: Animation Rendering
  - Platform: Heroku

- Results

- Conclusions and Future Work
Introduction
Introduction

- Master-Worker Architecture
  - Loosely-coupled parallel tasks
  - Centralized queue
  - MTC Applications, Batch Systems, Parallel Workflows, Map Reduce

- Limitations:
  - Single point of failure
  - Scalability
Platform as a Service (PaaS) Clouds
- Basic software stack – OS, run-time, DB, Logging
- Host you own application in Container
- On demand scaling
- Pay-as-you-go
- Limitations:
  - Ephemeral storage
  - Volatility
Motivation

- Migrate Master-Worker Applications to PaaS
  - Indirect communication
    - Message Queues
  - Move global state information outside container
Methodology
Agent-based Master-Worker

- Global Information in Message Queue
- Simple Reflex Agent
- Dynamically switch between Master/Worker
Master-Worker on PaaS
Agent-based Master-Worker

Start

Are tasks available?  
Yes: Fetch and Process a task  
No: Are all tasks finished?  
Yes: Compile and Store Results  
No: Timed Wait  

Is there a Master?  
Yes: Gain control and Push tasks to Message Queue  
No: Is there more work?  
Yes: Stop  
No: Start
Application: Animation Rendering

- High-fidelity Rendering of Animations
  - Image generation from physically-based quantities
- Computationally Intensive
  - Monte-Carlo Process
  - Loosely-coupled Parallel Tasks
- Many-Task Computing Application
Platform: Heroku

- Production Environment
- Support for long asynchronous jobs
- Application container – Dyno
  - Cost - $0.05/Hour
- Addon services
  - MongoDB and RabbitMQ
- Hosted on Amazon EC2
## Workloads

- **Animation Rendering (W1, W2)**
  - 800 x 600 Frame
  - 200 Tasks per Frame
  - Path Tracing

- **Nth Prime Number Calculation (W3)**
  - Five types

<table>
<thead>
<tr>
<th></th>
<th>Min Task Time (in s)</th>
<th>Max Task Time (in s)</th>
<th>Total Time (in Hr)</th>
<th>Total Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>1</td>
<td>460</td>
<td>240</td>
<td>24,000</td>
</tr>
<tr>
<td>W2</td>
<td>20</td>
<td>220</td>
<td>400</td>
<td>18,000</td>
</tr>
<tr>
<td>W3</td>
<td>20</td>
<td>60</td>
<td>190</td>
<td>18,000</td>
</tr>
</tbody>
</table>
Results
Speed-up

![Graph showing speed-up against number of dynos for different workloads (W1, W1-AM, W2, W2-AM, W3, W3-AM) and ideal speedup. The graph illustrates the trend and performance comparison across varying numbers of dynos.]
Efficiency: Wall-time
Efficiency: Actual Dyno-time
Ramp-up Times

![Graph showing ramp-up times for different numbers of dynos]

- **Y-axis**: Ramp-up Time (in s on log2 scale)
- **X-axis**: Number of Dynos (log2 scale)
- **Legend**:
  - W1
  - W2
  - W3
Task Execution Times: W3 Type 2

![Graph showing task execution times for different task instances and number of dynos.](image-url)
Task Execution Times: W3

[Graphs showing mean times and standard deviations for different types of tasks across dynos.]
Conclusion and Future Work

- Novel Master-Worker Architecture for PaaS
- 400 Hr workload in 52 minutes
  - 90% Wall-time Efficiency
- Effects of Multi-tenant PaaS infrastructure
- PaaS suitability for Master-Worker

- Alternate Message-brokers/Platforms
- Extension for Map Reduce
Thank You

- Contact:
  - Vibhor.Aggarwal@accenture.com

- Acknowledgements
  - Marko Dabrovic for Sponza Model
  - Heroku Engineers