Team D Project Plan

Cory Kolbeck, Tony Wooster, Erik Swanson, Adrian Miranda, Justin Wagner, and Federico Saldarini

Portland State University
Department of Computer Science
Portland, Oregon

June 9, 2011

Overview and Deliverables

We are to implement client libraries to provide asynchronous communication with a Burrow message queue server.

The code for these libraries will reside on Github, and the sponsor will link to them from the main Burrow site. Documentation will reside on the main Burrow website.

Assumptions

- The code will be maintained by the OpenStack community
- The code will continue to reside on Github, or possibly be moved to launchpad
- Documentation will be hosted on burrow.openstack.org
- Authentication is outside the scope of this project

Restraints

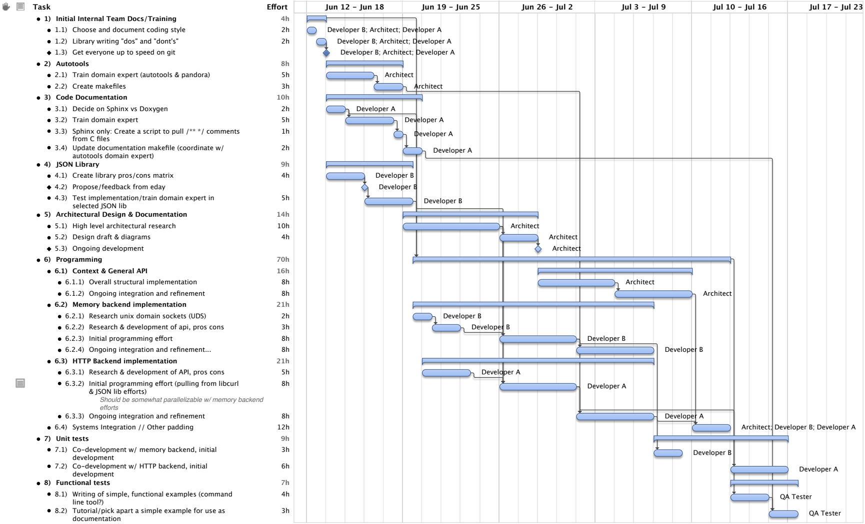
- Project will be distributed under the Apache2 license, and any libraries used must be compatible.
- All calls in to our library must be nonblocking.
- Maven will be used for Java builds.
- The Pandora autoconf macro set will be used for C builds.
- Minimal dependence on external libraries.

Plan Overview

The team will be split into two teams of three. Erik, Justin, and Cory will create a library in Java. Tony, Fede and Adrian will create a library in C.

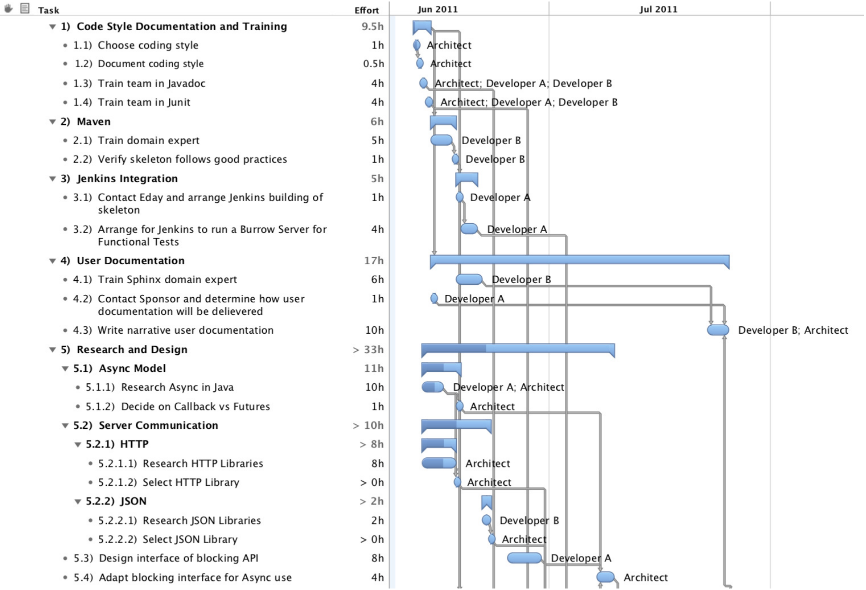
1. ~~Meet with Eric Day to discuss project details~~
2. ~~Decide on target languages~~
3. ~~Create skeleton projects~~ and integrate them with burrow continuous integration infrastructure
4. Create blocking memory backends
5. Research and choose JSON and HTTP libraries
6. Create blocking memory backend
7. Research asynchronous I/O
8. ~~Choose language appropriate callback mechanisms~~
9. Write asynchronous memory backend
10. Write asynchronous http backend
11. Write functional tests
12. *Time Allowing* Write small projects which use our libraries in interesting ways.

C Gantt Chart



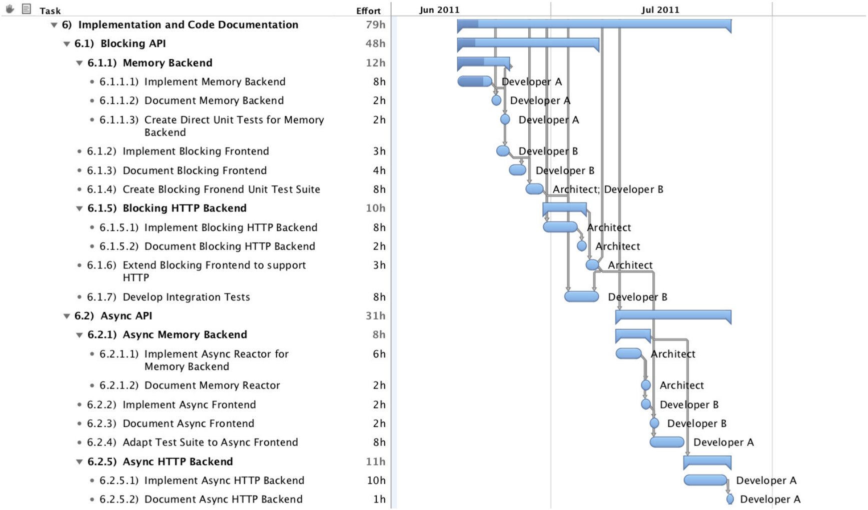
Task	Effort	
1) Initial Internal Team Docs/Training	4h	
1.1) Choose and document coding style	2h	Developer B; Architect; Developer A
1.2) Library writing "dos" and "dont's"	2h	Developer B; Architect; Developer A
1.3) Get everyone up to speed on git		Developer B; Architect; Developer A
2) Autotools	8h	
2.1) Train domain expert (autotools & pandora)	5h	Architect
2.2) Create makefiles	3h	Architect
3) Code Documentation	10h	
3.1) Decide on Sphinx vs Doxygen	2h	Developer A
3.2) Train domain expert	5h	Developer A
3.3) Sphinx only: Create a script to pull /** */ comments from C files	1h	Developer A
3.4) Update documentation makefile (coordinate w/ autotools domain expert)	2h	Developer A
4) JSON Library	9h	
4.1) Create library pros/cons matrix	4h	Developer B
4.2) Propose/feedback from eday		Developer B
4.3) Test implementation/train domain expert in selected JSON lib	5h	Developer B
5) Architectural Design & Documentation	14h	
5.1) High level architectural research	10h	Architect
5.2) Design draft & diagrams	4h	Architect
5.3) Ongoing development		Architect
6) Programming	70h	
6.1) Context & General API	16h	
6.1.1) Overall structural implementation	8h	Architect
6.1.2) Ongoing integration and refinement	8h	Architect
6.2) Memory backend implementation	21h	
6.2.1) Research unix domain sockets (UDS)	2h	Developer B
6.2.2) Research & development of api, pros cons	3h	Developer B
6.2.3) Initial programming effort	8h	Developer B
6.2.4) Ongoing integration and refinement...	8h	Developer B
6.3) HTTP Backend implementation	21h	
6.3.1) Research & development of API, pros cons	5h	Developer A
6.3.2) Initial programming effort (pulling from libcurl & JSON lib efforts) *Should be somewhat parallelizable w/ memory backend efforts*	8h	Developer A
6.3.3) Ongoing integration and refinement	8h	Developer A
6.4) Systems Integration // Other padding	12h	Architect; Developer B; Developer A
7) Unit tests	9h	
7.1) Co-development w/ memory backend, initial development	3h	Developer B
7.2) Co-development w/ HTTP backend, initial development	6h	Developer A
8) Functional tests	7h	
8.1) Writing of simple, functional examples (command line tool?)	4h	QA Tester
8.2) Tutorial/pick apart a simple example for use as documentation	3h	QA Tester

Java Gantt Chart



Task	Effort	Jun 2011	Jul 2011
▼ 1) **Code Style Documentation and Training**	9.5h		
• 1.1) Choose coding style	1h	Architect	
• 1.2) Document coding style	0.5h	Architect	
• 1.3) Train team in Javadoc	4h	Architect; Developer A; Developer B	
• 1.4) Train team in Junit	4h	Architect; Developer A; Developer B	
▼ 2) **Maven**	6h		
• 2.1) Train domain expert	5h	Developer B	
• 2.2) Verify skeleton follows good practices	1h	Developer B	
▼ 3) **Jenkins Integration**	5h		
• 3.1) Contact Eday and arrange Jenkins building of skeleton	1h	Developer A	
• 3.2) Arrange for Jenkins to run a Burrow Server for Functional Tests	4h	Developer A	
▼ 4) **User Documentation**	17h		
• 4.1) Train Sphinx domain expert	6h	Developer B	
• 4.2) Contact Sponsor and determine how user documentation will be delievered	1h	Developer A	
• 4.3) Write narrative user documentation	10h		Developer B; Architect
▼ 5) **Research and Design**	> 33h		
▼ 5.1) **Async Model**	11h		
• 5.1.1) Research Async in Java	10h	Developer A; Architect	
• 5.1.2) Decide on Callback vs Futures	1h	Architect	
▼ 5.2) **Server Communication**	> 10h		
▼ 5.2.1) **HTTP**	> 8h		
• 5.2.1.1) Research HTTP Libraries	8h	Architect	
• 5.2.1.2) Select HTTP Library	> 0h	Architect	
▼ 5.2.2) **JSON**	> 2h		
• 5.2.2.1) Research JSON Libraries	2h	Developer B	
• 5.2.2.2) Select JSON Library	> 0h	Architect	
• 5.3) Design interface of blocking API	8h	Developer A	
• 5.4) Adapt blocking interface for Async use	4h		Architect

Java Gantt Chart



Task	Effort
6) Implementation and Code Documentation	79h
6.1) Blocking API	48h
6.1.1) Memory Backend	12h
6.1.1.1) Implement Memory Backend	8h
6.1.1.2) Document Memory Backend	2h
6.1.1.3) Create Direct Unit Tests for Memory Backend	2h
6.1.2) Implement Blocking Frontend	3h
6.1.3) Document Blocking Frontend	4h
6.1.4) Create Blocking Fronend Unit Test Suite	8h
6.1.5) Blocking HTTP Backend	10h
6.1.5.1) Implement Blocking HTTP Backend	8h
6.1.5.2) Document Blocking HTTP Backend	2h
6.1.6) Extend Blocking Frontend to support HTTP	3h
6.1.7) Develop Integration Tests	8h
6.2) Async API	31h
6.2.1) Async Memory Backend	8h
6.2.1.1) Implement Async Reactor for Memory Backend	6h
6.2.1.2) Document Memory Reactor	2h
6.2.2) Implement Async Frontend	2h
6.2.3) Document Async Frontend	2h
6.2.4) Adapt Test Suite to Async Frontend	8h
6.2.5) Async HTTP Backend	11h
6.2.5.1) Implement Async HTTP Backend	10h
6.2.5.2) Document Async HTTP Backend	1h

Calendar

Week of	*Deliverable*
Jun 05	Project Plan, Architecture Overview
Jun 12	
Jun 19	
Jun 26	Blocking Memory Backend
Jul 03	Blocking HTTP Backend
Jul 10	
Jul 17	Async Memory Backend
Jul 24	Async HTTP Backend
Jul 31	Narrative Documentation, Sponsor Delivery
Aug 08	Final Presentation

Meetings and Reviews

- In person meetings with sponsor every 1-2 weeks.
- IRC consultation as needed.
- Reviews at milestones as previously noted.

Resource Identification - Time

Name	*Available Hours/Week*
Justin	8-10
Adrian	8-10
Tony	8-10
Erik	10-15
Federico	10+
Cory	10-15

Resource Identification - Expertise

Name	*Expertise*
Justin	-
Adrian	-
Tony	REST
Erik	REST
Federico	-
Cory	Parallel architecture, Basic Git
Eric	Async I/O in C, Burrow

Configuration Management

- Github will be used for source control
- Ticketing and bug reporting will be through Github's builtin utilities
- Should it become necessary, language leads will be in charge of resolving merge conflicts
- Unit testing and commit screening will be will be through the Jenkins continuous integration framework.

Roles

Role	*Responsibility*	Initial
Manager	Coordinate general geetings and maintain schedules	C
POC	Maintain communication between team and sponsor	C
Integration	Support Jenkins and Github issues	C
Java Lead	Architect Java library and delegate coding and research tasks	E
C Lead	Architect C library and delegate coding and research tasks	T
Java Dev	Implement designs of Java lead	CJE
C Dev	Implement designs of C lead	AFT
Support	Maintain Shared Machines	C
Unit Testing	Code unit tests for every function	All
Func. Testing	Create functional tests for each language	tbd
API Docs	Write language specific documentation	tbd
General Docs	Create a language agnostic guide to coding burrow clients	T

Risk Management

- Risk: Team member drops out
  Consequence: Fewer man-hours available
  Mitigation: Scheduling as if we have less time
- Risk: Architect drops out Consequence: Possible loss of grand plan Mitigation: Documentation, regular meetings to keep team members in the loop
- Risk: API Change mid-project
  Consequence: Project may need re-architecting
  Mitigation: Modular design

QA and Deployment

- Unit and regression testing will be ongoing using Burrow's existing Jenkins continuous integration system.
- Functional testing will take place in the weeks leading up to code freeze.
- Deployment will consist of linking to our existing Github repositories (or possibly official forks) from burrow.openstack.org.
- Documentation will be pulled from Github by our sponsor for inclusion in the Burrow wiki.