

Build Service

Requirements Document

Author: Steve Jones

Date: 4/18/2014

The purpose of this project is to design and implement a system for regular (nightly or other experiment-level) software builds by Frontier experiments and related software providers at Fermilab. Presently, many software packages are built (on a nightly basis) on interactive nodes. While this is easy to set up, the builds take a long time—up to many hours. Limitations include I/O bandwidth, e.g., from use of network-attached storage, such as NFS or AFS; and a limited number of processors/cores, which limits parallelism in the build process. Individual users building code for their own analyses face similar problems, long compile and link times, probably for the same reasons.

Table of Contents

I. Executive Summary	3
II. Requirements Summary	3
III. Assumptions, Risks, Dependencies	7
IV. Out of Scope.....	7
V. Performance and Key Success Metrics.....	7
VI. Use Cases.....	8
VII. Architecture	9
VIII. Detailed Functional and System Requirements	9
IX. Detailed Business Process Flow Diagrams	10
X. Reports	11
XI. Stakeholders	11
XII. Project Team	12
XIII. Revision History	12

I. Executive Summary

The purpose of this project is to design and implement a system for regular (nightly or other experiment-level) software builds by Frontier experiments and related software providers at Fermilab.

Presently, many software packages are built (on a nightly basis) on interactive nodes. While this is easy to set up, the builds take a long time—up to many hours. Limitations include I/O bandwidth, e.g., from use of network-attached storage, such as NFS or AFS; and a limited number of processors/cores, which limits parallelism in the build process.

Individual users building code for their own analyses face similar problems, long compile and link times, probably for the same reasons.

A related problem is that the build process is not well integrated with code distribution, including CVMFS.

The build system architecture from this project should enable greatly reduced build times: tens of minutes or less, rather than hours. The system should be reasonably easy for experiment developers and software librarians to use; affordable within expected budget constraints; and maintainable without undue expense or administration effort. Support for remote build machines is considered in scope for this project.

II. Requirements Summary

Provide a high level list of the requirements for this project.

No.	Requirement	Category	Source	Priority
1.	Hardware:	Hardware	Glenn Cooper	High
1.a	Memory requirements are modest, 2 GB/core are sufficient	Hardware	Glenn Cooper	High
1.b	Begin with a few 16-core systems	Hardware	Glenn Cooper	High
1.c	Need at least one running SLF5 and one running SLF6	Hardware	Glenn Cooper	High
1.d	Can add other platforms (Mac; Ubuntu, SUSE; ARM; ...) later	Hardware	Glenn Cooper	High
1.e	Few TB local disk	Hardware	Glenn Cooper	High
2.	A survey of existing solutions must be performed and a report that presents the arguments for a choice must be drafted. Free Software or Open Source solutions must be considered first and proprietary systems only	Operational	Glenn Cooper/Brett Viren LBNE/LArSoft	High

	considered if no FOSS solutions are suitable. The Framework should support:			
2.a	Continuous integration is desired.	Operational	Glenn Cooper	High
2.b	The system must retain an association between a job run and a particular state (commit) of the repository holding the main software being tested.	Operational	Brett Viren from LBNE/LArSoft	High
2.c	Incremental and green-field building of the entire experiment software stack from source	Operational	Brett Viren from LBNE/LArSoft	High
2.d	Analysis that compare current output to prior output including log files with transient changes filtered and histograms.	Operational	Brett Viren from LBNE/LArSoft	High
2.e				
2.f				
2.g	Build Service must accept remote and/or manual trigger.	Operational	3/28/14 meeting	High
2.h	A job that is run must be recorded based on unique metadata including: target host, associated version (git commit hash, svn revision number), job domain (eg, package or test name).	Operational	Brett Viren from LBNE/LArSoft	High
2.i	Success and failure reports should trigger email notifications to an opt-in list.	Operational	Brett Viren from LBNE/LArSoft	High
2.j	Jobs must be able to run on all supported platforms.	Operational	Brett Viren from LBNE/LArSoft	High
2.j.1	Service must not constrain platforms; platforms must be able to run the job and contribute results but possibly with additional effort provided.	Operational	Brett Viren from LBNE/LArSoft; 3/28/14 meeting	High
2.j.2	Job processes must be able to run on hosts on non-Fermilab networks which may be behind	Operational	Brett Viren from LBNE/LArSoft	High

	firewalls with default-deny for incoming connections.			
2.j.3	It must be possible to invalidate any given job result in order to trigger it to rerun.	Operational	Brett Viren from LBNE/LArSoft	High
2.j.4	System should allow user to view errors in jobs run and provide links to access the file system of jobs	Operational	3/21/Meeting	Low
3.	Reporting:	Operational	Glenn Cooper	High
3.a	Provide current status of each job	Operational	Glenn Cooper	High
3.b	Show the success/failure of completed jobs	Operational	Glenn Cooper	High
3.c	Show resources used	Operational	Glenn Cooper	High
3.d	Report on job results be they success and failure must be stored and made available for browsing via the web.	Operational	Brett Viren from LBNE/LArSoft	High
3.e	Provide report that should indicate what triggered the job. (possibly interfacing with Redmine)	Operational	Brett Viren from LBNE/LArSoft	High
3.f	Historical success/failure rates of builds	Operational	3/21/Meeting	High
3.g	Days since last successful/unsuccessful build for each slave	Operational	3/21/Meeting	High
4.	Be robust enough to be able to support the number of potential participants (experiments, projects)	Operational	Glenn Cooper	High
4.a	IF experiments: 10 (g-2, LBNE, MicroBooNE, MINERvA, MINOS, Mu2E, NOvA, SciBooNE, SeaQuest)	Operational	Glenn Cooper	High
4.b	CF experiments: 3 (DarkSide, DES, LSST)	Operational	Glenn Cooper	High
4.c	Software projects: 2 (LArSoft,	Operational	Glenn Cooper	High

	art)			
4.d	Expandable for growth for 10 additional future participants	Operational	Steve Jones	Medium
5.	Builds must support working with offsite hardware (eg BNL).	Operational	Brett Viren from LBNE/LArSoft	High
6.	Terminology: <ol style="list-style-type: none"> 1. Build Platform(s): includes OS version; compiler version; optionally other details 2. Build Service: monitor, coordinate; support build for remote sites; works with meta data for each slave; basically integration service 3. Build Framework is the continuous integration or build automation software 4. Build Master = Service Machine 5. Build Slave = Platform Machine 	Operational	3/14 and 3/21 Meeting	High
7.	Security and Access Requirements	Operational	3/14 Meeting	High
7.a	System must provide user levels to create new jobs, to run jobs and to access reports on jobs	Operational	3/21/Meeting	High
8.	Documentation Requirements	Operational	3/14 Meeting	High
8.1	Prior to initial deployment, documentation for expert users in the experiments must be provided (eg. Wiki Users Guide)	Operational	3/21/Meeting	High
9.	Redmine Integration	Operational	3/14 Meeting	Low
9.1	Build failures trigger bug report	Operational	3/21/Meeting	Low
9.2	Build reports stored on both Master server and Redmine	Operational	3/21/Meeting	Low
9.3	Redmine provides status of	Operational	3/21/Meeting	Low

	Master			
9.4	Build project history stored on Redmine	Operational	3/21/Meeting	Low
10.	Build process needs to be integrated with Service Strategy/Service Design	Operational	Steve Jones/ Mike Kaiser	High

III. Assumptions, Risks, Dependencies

1. There are several options for frameworks that need to be evaluated. Options include, in very rough order of interest expressed:
 - BuildBot (Python based; used by MINERvA)
 - Jenkins (Java based; used by CMS, LHCb)
 - Trac/Bitten/Nose (Python based; used by Daya Bay)
 - NICOS (shell scripts, Python; used by ATLAS, developed in house)
 - Condor or other batch system
 - Cron entries
 - Many others—this is by no means a comprehensive list.
2. Resources to conduct the project are available within currently assigned staff.
3. Project cost is level of effort only except for Project Management costs.
4. Hardware can be reallocated from existing sources or will be identified and procured within existing budgets.

IV. Out of Scope

1. The initial scope does not include a facility for individual experiment members to build/test their own analysis code. That could be considered in a later project or a new phase of this project.
2. The system does not perform software delivery functions; that will be done by other web services or applications like CVMFS

V. Performance and Key Success Metrics

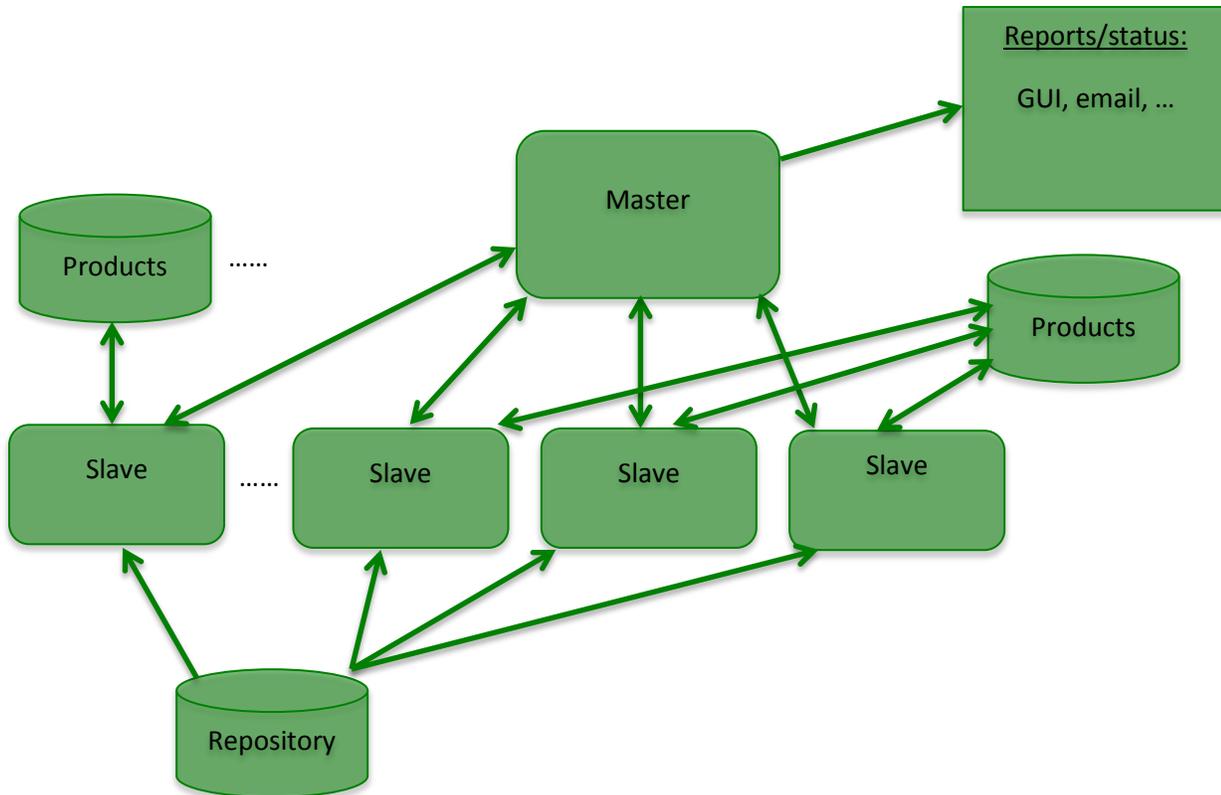
- Function
 - Build service architecture defined
 - Overall design of the build service completed:
 - Hardware elements identified
 - Mechanisms to schedule builds: software package, batch system, etc. defined
 - Hardware procurement and installation (if new hardware is required) complete
 - Build system configuration and testing completed
 - Framework for supervision and administration of the build service deployed
 - Transaction Throughput sufficient for users
 - Batch Throughput sufficient for users
- Users
 - IF experiments: 10 (, g-2, LBNE, MicroBooNE, MINERvA, MINOS, Mu2E, NOvA, SciBooNE, SeaQuest), CF experiments: 3 (DarkSide, DES, LSST) and Software projects: 2 (LArSoft, art) and potential future users are able to simultaneous build each night

VI. Use Cases

1. Nightly (or other periodic) code build
 - a. Actors: an experiment, a project, or a major component of one
 - b. Schedule is set on master
 - c. May also include unit tests, validation modules; or these can be separate
 - d. Collects changes made over specified period
2. Continuous integration
 - a. Jobs launched by [particular classes of] code check-ins; time intervals; or other triggers
 - b. Typically runs unit tests along with each build
3. Build for additional platforms
 - a. "Platform" may include OS version; compiler version; hardware type; etc.
 - b. Master can send jobs to any platform; build requires only a slave with the desired characteristics
4. Manual (aperiodic) code build
 - a. Actors: an experiment, a project, or a major component of one
 - b. Input manually on master
 - c. May also include unit tests, validation modules; or these can be separate

VII. Architecture

The figure below shows a high-level view of the build system to be implemented:



Build automation software running on the master schedules build, test, and validation jobs. To start a scheduled job, the master selects a slave and sends scripts or other information to the slave. On the slave, the job pulls source code and other data, if any, from one or more repositories; for an incremental build or validation, the slave may also pull from the output “products” area to get the previous result. The master keeps track of the status of each job on each slave, visible typically via a web interface, email notifications, and other mechanisms. When a job completes, it sends status and other metadata to the master, and copies its products—built binaries, test results, or other information—to specified destinations.

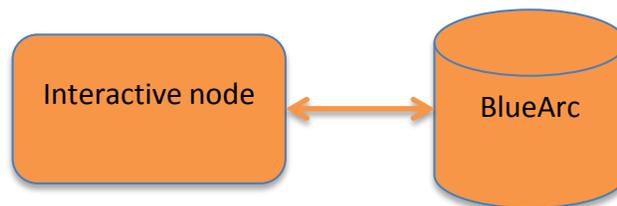
VIII. Detailed Technical and System Requirements

1. Build automation software (or continuous integration software)
 - a. Runs on master
 - b. Must be maintainable and supportable with minimum level of effort
 - c. Process for users to schedule/submit jobs must be simple
 - d. Client software must be supported on all required platforms (see user requirements)
 - e. Needs a way to trigger a rebuild, i.e., repeat a build
 - f. Must track and make viewable:
 - i. Job status

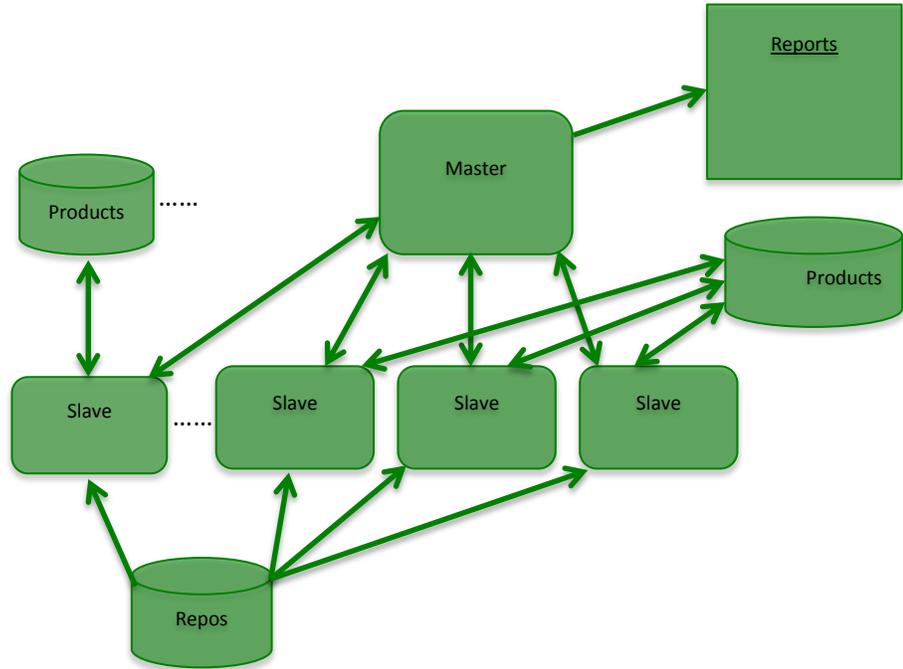
- ii. Success/failure of each job
 - iii. Resources used by each job
 - iv. Historical records of ii and iii
 - g. Must support submissions by multiple users
 - h. Must be scalable to allow for growth in number of experiments/projects and in number of builds by each
 - i. Must be able to send jobs to slaves both at Fermilab and at other locations
 - j. Open source preferred
2. Hardware
 - a. Master
 - i. Needs modest CPU, RAM, local storage
 - ii. Suitable for a VM
 - iii. Will run Scientific Linux
 - b. Slaves
 - i. For frequent builds, need multiple cores, at least modest local storage
 - ii. At least one slave of each required hardware type; but some can be remote
 - iii. For infrequent builds, could use smaller (physical or virtual) systems
 - c. Additional platforms for logins and interactive builds?

IX. Detailed Business Process Flow Diagrams

- Most common current method: Log in, build/test/validate on same node. Source code read from network-attached storage; results written to network-attached storage.



- Central build system design: Jobs sent from master to slaves with local storage; see Architecture section above.



X. Reports

- Current status of each job
- Success/failure of completed jobs
- Historical success/failure of jobs
- Resources used
- Days since last successful/unsuccessful build
- What triggered job

XI. Stakeholders

Group	Name	Role
SCD	Ruth Pordes, Stu Feuss	Sponsors
FEF	Stu Feuss	Implementation Owner
NOvA	Andrew John Norman	Users
Minerva	Erica Snider	Users
LBNE	Eileen F. Berman, Qizhong Li, Brett Viren	Users
Microboone	Stephen A. Wolbers	Users
Muon g-2	Adam L. Lyon	Users
Darkside 50	Kenneth Richard Herner	Users
Minos	Arthur E. Kreymer	Users
SciBoone		Users
SeaQuest		Users
DES		Users
LSST		Users
LARsoft	Ruth Pordes, Erica Snider	Users
art	Jim Kowalkowski, Chris Green	Users
Mu2e	Rob Kutschke	Users

XII. Project Team

Name	Role
Steve Jones	Project Manager
Glenn Cooper	Architect/ Project Technical Lead
Ed Simmonds	Assistant Technical Lead
Liz Sexton-Kennedy	Assistant Technical Lead
Marc Mengel	Developer
Patrick Gartung	Developer
Seth Graham	Developer

XIII. Revision History

Version	Date	Author	Notes
0.1	3/12/14	Steve Jones	Initial draft
0.2	3/12/14	Steve Jones	Revision based on inputs from LBNE
0.3	3/17/14	Steve Jones	Revision after meeting with SCD managers
0.4	3/19/14	Steve Jones	Revision based on Ruth and Glenn notes
0.5	3/21/14	Steve Jones	Revision based on today's meeting and Glenn's technical requirements
0.6	3/28/14	Steve Jones	Revision based on today's meeting
0.7	4/11/14	Steve Jones	Accepted changes and revision based on today's meeting
0.8	4/17/14	Glenn Cooper	Added diagrams and other material
0.9	4/18/14	Steve Jones	Finalize per today's meeting and send to Liaisons for approval