

Update on Testing Infrastructure and SVN statistics

Sergey Lyskov

Jeff GrayLab JHU
2011

<http://rosettatests.graylab.jhu.edu>



Tests Statistics



Score Function Fingerprint



SVN Statistics



Tests Statistics

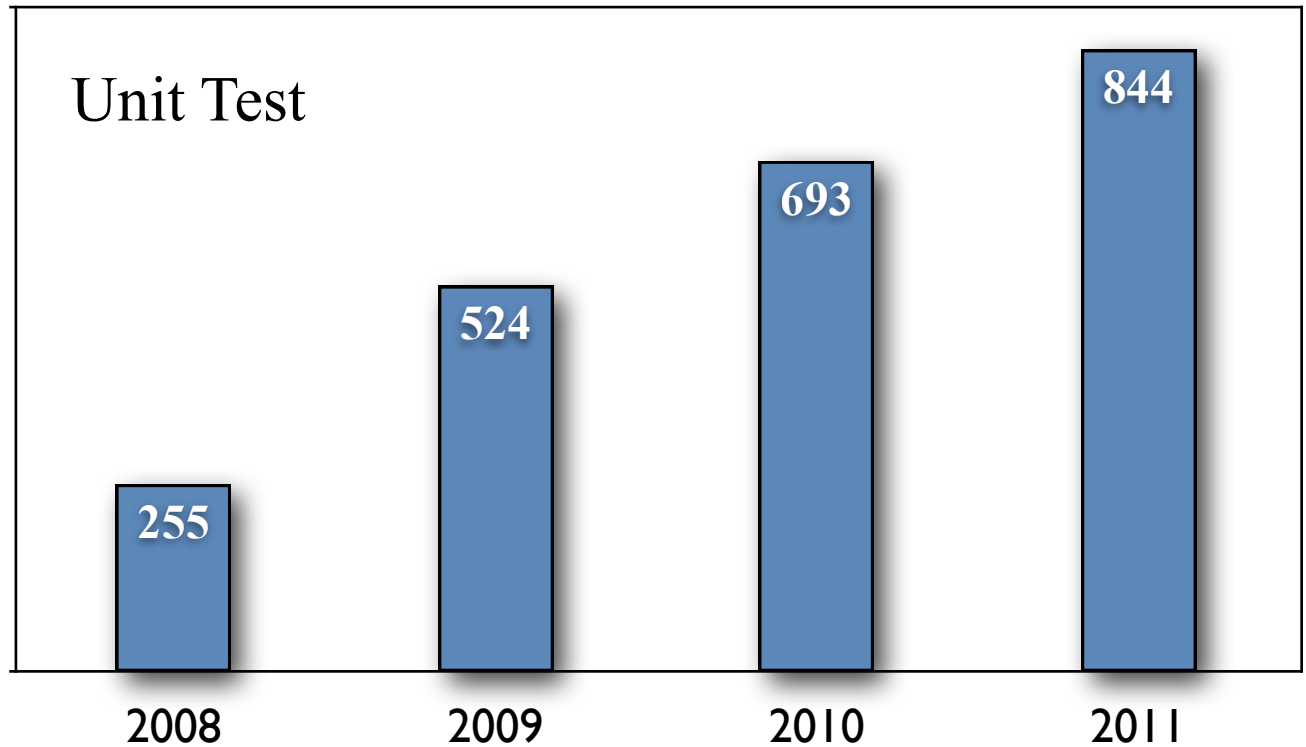
Quick overview, the types of tests we currently run:

Quick overview, the types of tests we currently run:

- Build tests

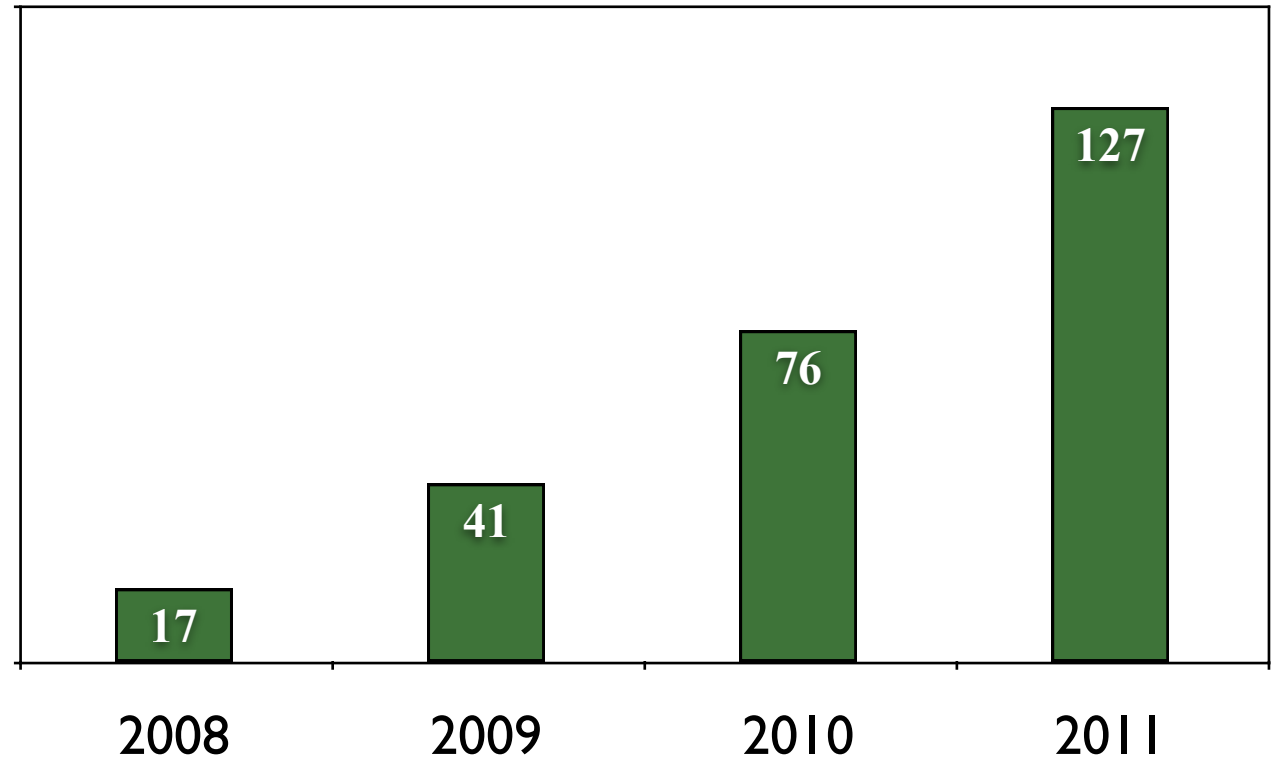
Quick overview, the types of tests we currently run:

- Build tests
- Unit tests



Quick overview, the types of tests we currently run:

- Build tests
- Unit tests
- Integration tests



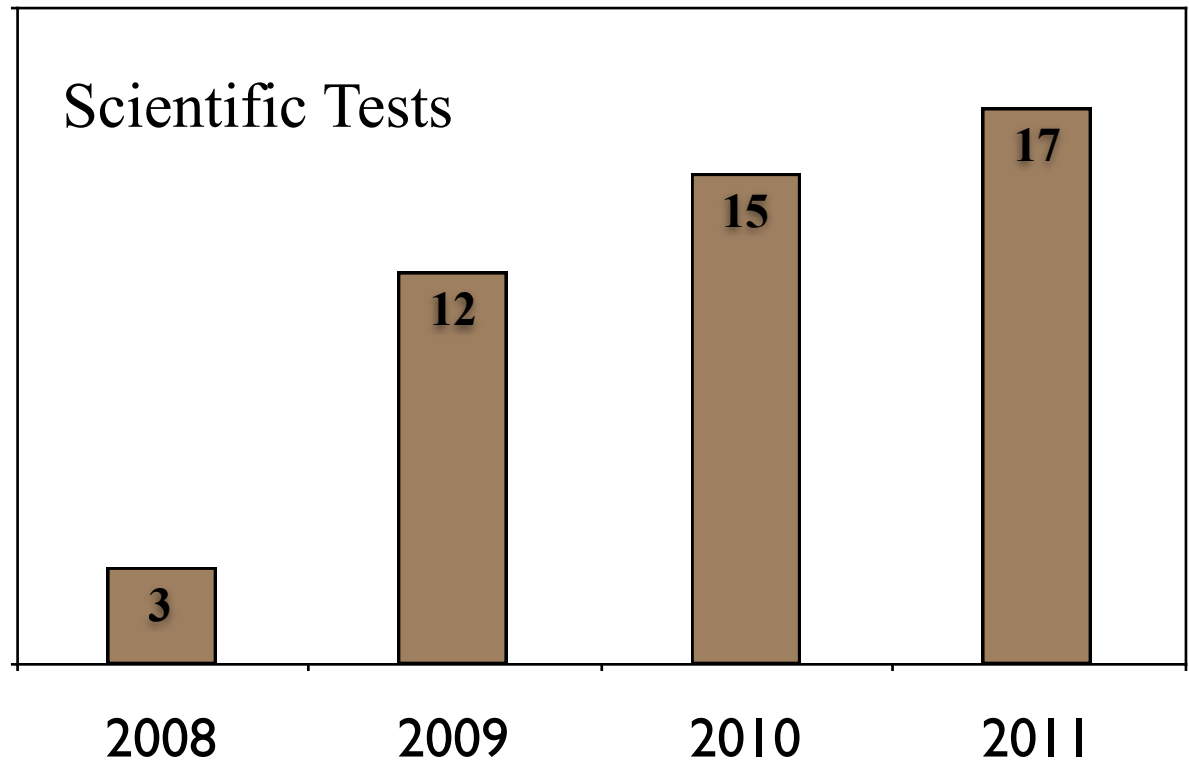
Quick overview, the types of tests we currently run:

- Build tests
- Unit tests
- Integration tests
- Performance tests
- Profile test
- [new this year] Score Function fingerprint

| | 2010 | 2011 |
|----------------------------------|------|------|
| Profile Tests | 9 | 11 |
| Performance Tests | 9 | 8 |
| Score Function fingerprint Tests | - | 3 |

Quick overview, the types of tests we currently run:

- Build tests
- Unit tests
- Integration tests
- Performance tests
- Profile test
- [new this year] Score Function fingerprint
- Scientific tests



Few new scientific tests added. Why?

Few new scientific tests added. Why?

- Scientific tests are added *after* protocol publication. But by then the developer is about to leave!

Few new scientific tests added. Why?

- Scientific tests are added *after* protocol publication. But by then the developer is about to leave!
 - Suggestion: Make ST addition a ‘publishing requirement’ (as we do for doc’s)

Few new scientific tests added. Why?

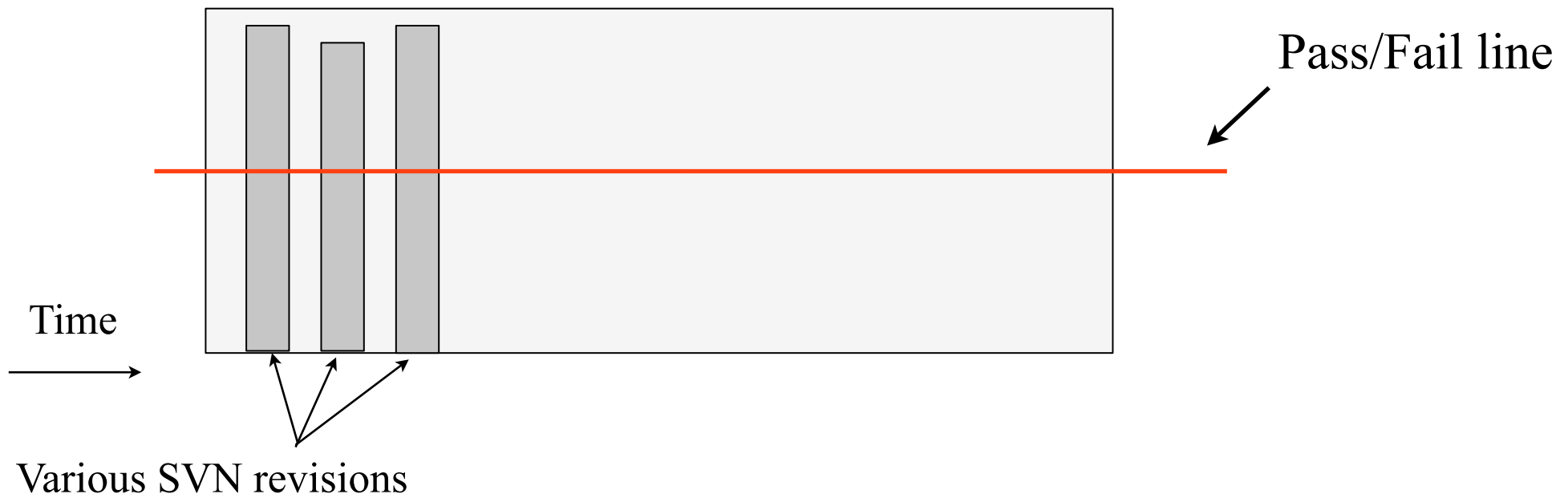
- Scientific tests are added *after* protocol publication. But by then the developer is about to leave!
 - Suggestion: Make ST addition a ‘publishing requirement’ (as we do for doc’s)
 - Use ST as a developer tool as opposed to a test for *finished* protocol

Few new scientific tests added. Why?

- Scientific tests are added *after* protocol publication. But by then the developer is about to leave!
 - Suggestion: Make ST addition a ‘publishing requirement’ (as we do for doc’s)
 - Use ST as a developer tool as opposed to a test for *finished* protocol
 - In that case, Pass/Fail ST output might not be the best thing:

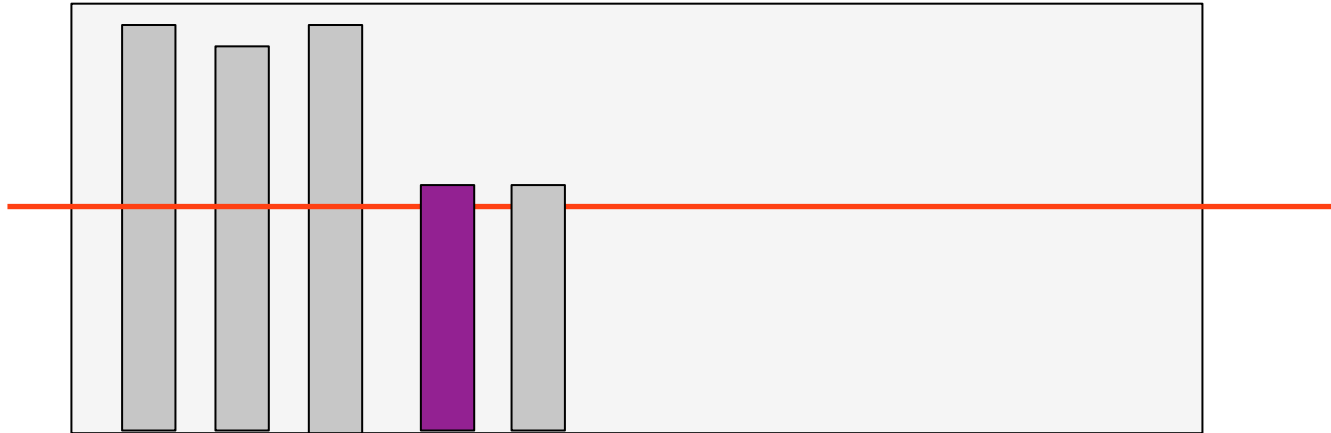
Few new scientific tests added. Why?

- Scientific tests are added *after* protocol publication. But by then the developer is about to leave!
 - Suggestion: Make ST addition a ‘publishing requirement’ (as we do for doc’s)
 - Use ST as a developer tool as opposed to a test for *finished* protocol
 - In that case, Pass/Fail ST output might not be the best thing:



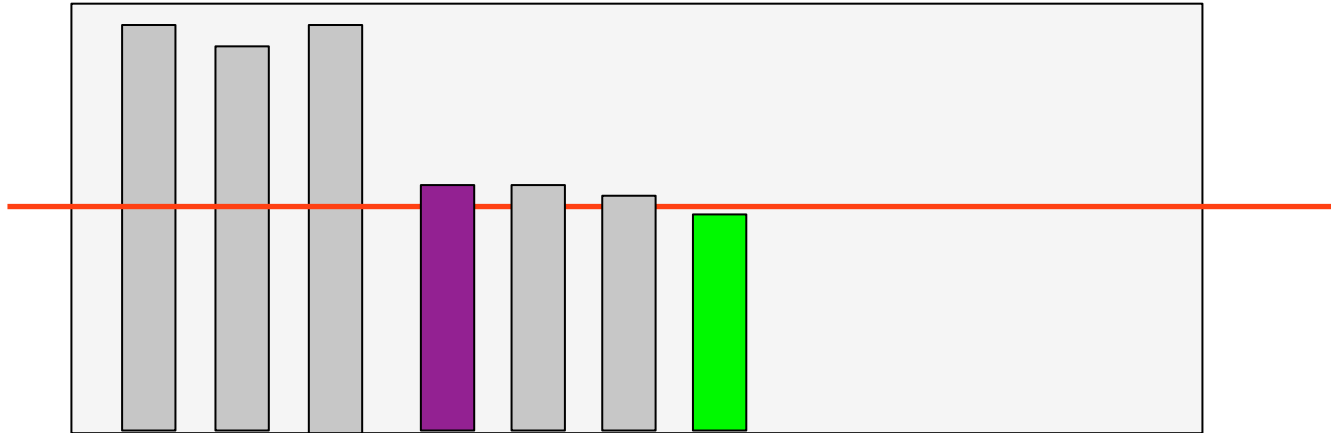
Few new scientific tests added. Why?

- Scientific tests are added *after* protocol publication. But by then the developer is about to leave!
 - Suggestion: Make ST addition a ‘publishing requirement’ (as we do for doc’s)
 - Use ST as a developer tool as opposed to a test for *finished* protocol
 - In that case, Pass/Fail ST output might not be the best thing:



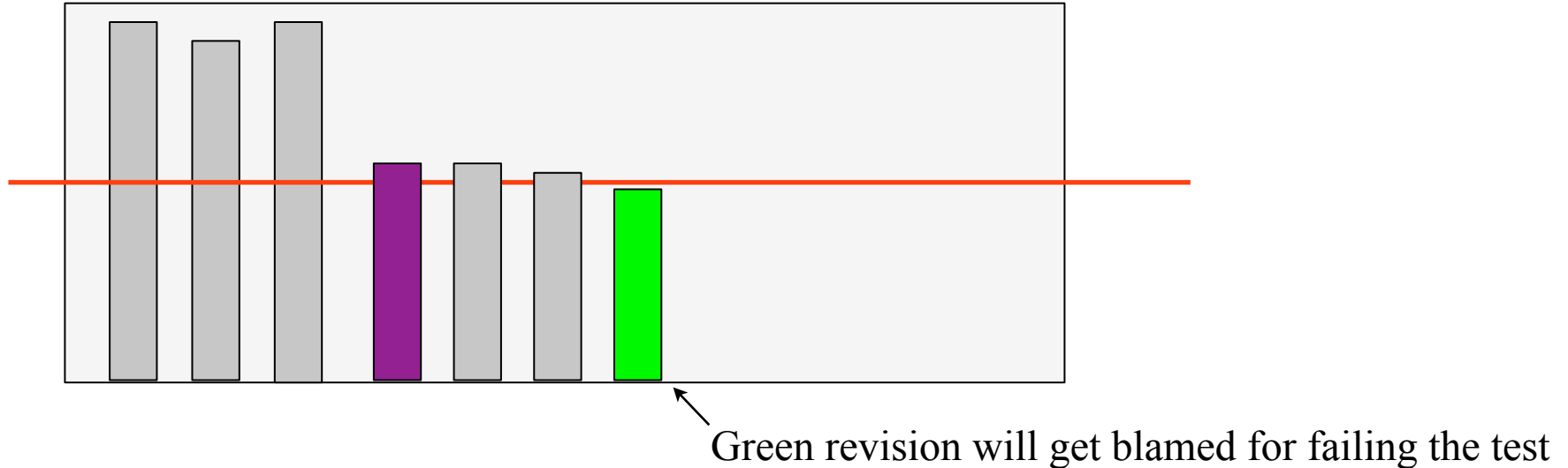
Few new scientific tests added. Why?

- Scientific tests are added *after* protocol publication. But by then the developer is about to leave!
 - Suggestion: Make ST addition a ‘publishing requirement’ (as we do for doc’s)
 - Use ST as a developer tool as opposed to a test for *finished* protocol
 - In that case, Pass/Fail ST output might not be the best thing:



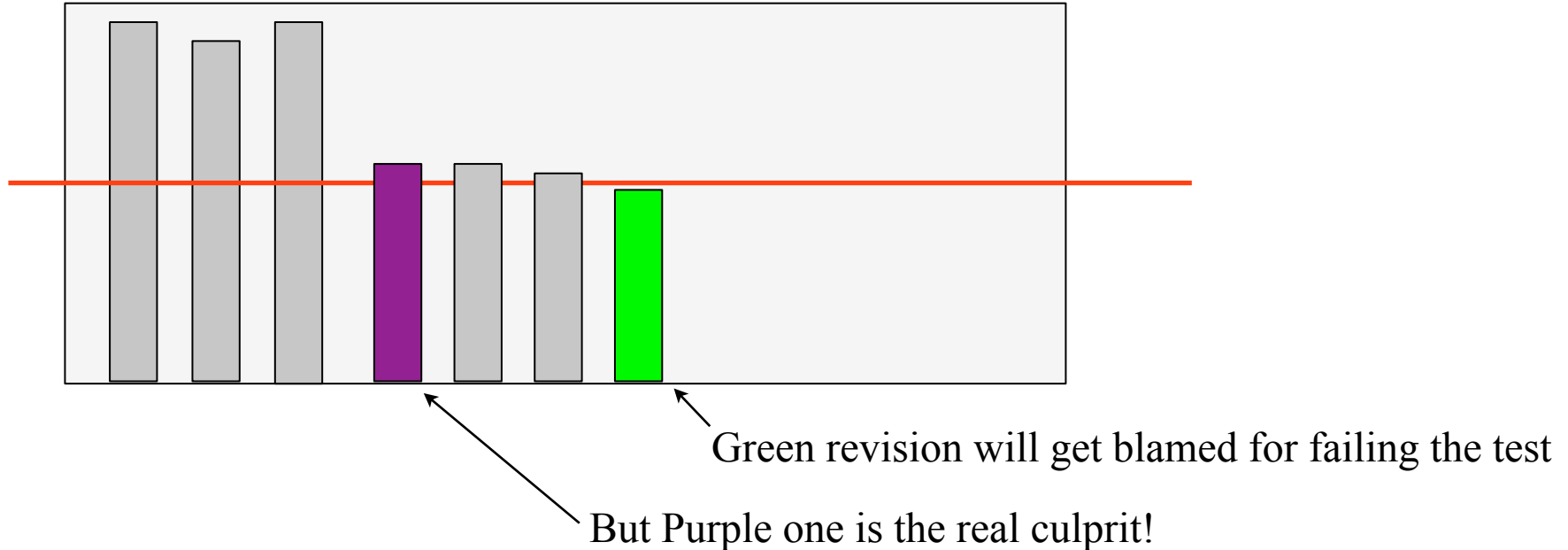
Few new scientific tests added. Why?

- Scientific tests are added *after* protocol publication. But by then the developer is about to leave!
 - Suggestion: Make ST addition a ‘publishing requirement’ (as we do for doc’s)
 - Use ST as a developer tool as opposed to a test for *finished* protocol
 - In that case, Pass/Fail ST output might not be the best thing:



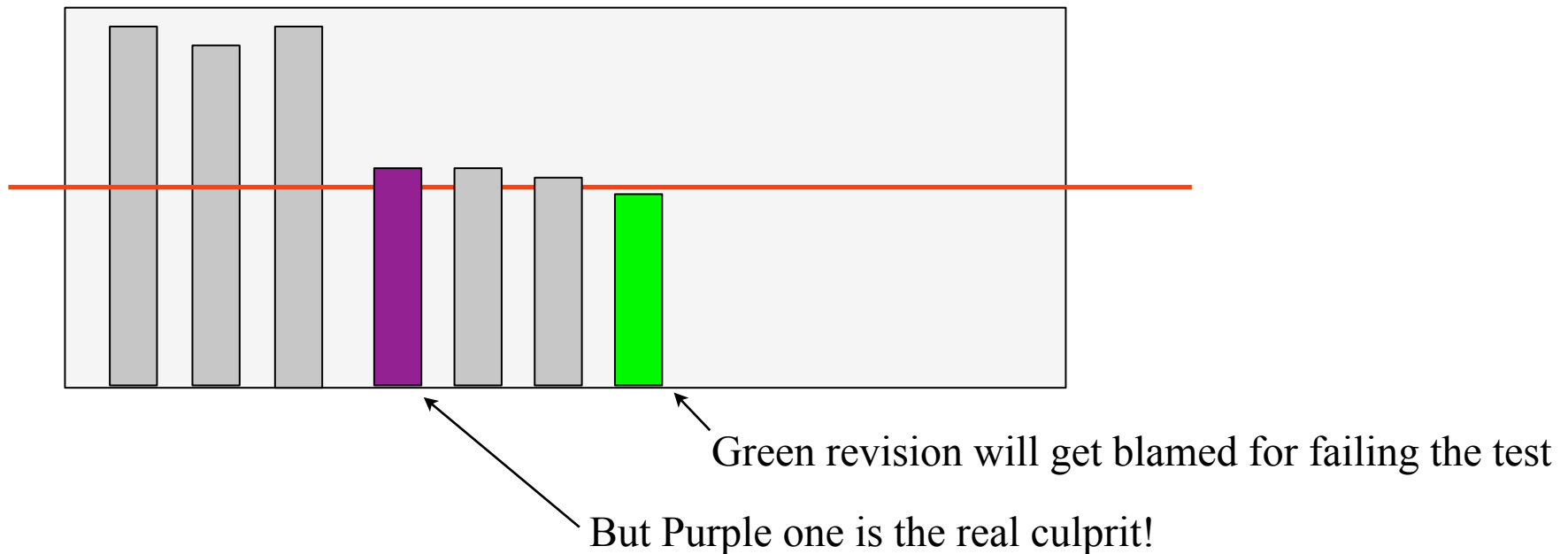
Few new scientific tests added. Why?

- Scientific tests are added *after* protocol publication. But by then the developer is about to leave!
 - Suggestion: Make ST addition a ‘publishing requirement’ (as we do for doc’s)
 - Use ST as a developer tool as opposed to a test for *finished* protocol
 - In that case, Pass/Fail ST output might not be the best thing:



Few new scientific tests added. Why?

- Scientific tests are added *after* protocol publication. But by then the developer is about to leave!
 - Suggestion: Make ST addition a ‘publishing requirement’ (as we do for doc’s)
 - Use ST as a developer tool as opposed to a test for *finished* protocol
 - In that case, Pass/Fail ST output might not be the best thing:



➡ So boolean result for scientific tests might not be our best choice...



Tests Statistics



Score Function Fingerprint

New test type: Score-Function-Fingerprint

New test type: Score-Function-Fingerprint

- Each test calls 'version_scorefunction' (written by Andrew).

New test type: Score-Function-Fingerprint

- Each test calls 'version_scorefunction' (written by Andrew).
- Testing server runs these tests for all Mini revisions and keeps track of differences.

New test type: Score-Function-Fingerprint

- Each test calls 'version_scorefunction' (written by Andrew).
- Testing server runs these tests for all Mini revisions and keeps track of differences.
- **Each score function is fully ID'd with unique number - its version** [equal to SVN revision which initially introduces it]

New test type: Score-Function-Fingerprint

- Each test calls 'version_scorefunction' (written by Andrew).
- Testing server runs these tests for all Mini revisions and keeps track of differences.
- **Each score function is fully ID'd with unique number - its version** [equal to SVN revision which initially introduces it]
- Comparison: right now literal, in near future with specified numeric precisions.

New test type: Score-Function-Fingerprint

- Each test calls 'version_scorefunction' (written by Andrew).
- Testing server runs these tests for all Mini revisions and keeps track of differences.
- **Each score function is fully ID'd with unique number - its version** [equal to SVN revision which initially introduces it]
- Comparison: right now literal, in near future with specified numeric precisions.
- Testing server will automatically send notification email when score fn changes. User can inspect history of changes using web interface similar to SVN-trac

New test type: Score-Function-Fingerprint

- Each test calls 'version_scorefunction' (written by Andrew).
- Testing server runs these tests for all Mini revisions and keeps track of differences.
- **Each score function is fully ID'd with unique number - its version** [equal to SVN revision which initially introduces it]
- Comparison: right now literal, in near future with specified numeric precisions.
- Testing server will automatically send notification email when score fn changes. User can inspect history of changes using web interface similar to SVN-trac
- Currently we have 4 of these tests implemented; additions welcome!

New test type: Score-Function-Fingerprint

- Each test calls 'version_scorefunction' (written by Andrew).
- Testing server runs these tests for all Mini revisions and keeps track of differences.
- **Each score function is fully ID'd with unique number - its version** [equal to SVN revision which initially introduces it]
- Comparison: right now literal, in near future with specified numeric precisions.
- Testing server will automatically send notification email when score fn changes. User can inspect history of changes using web interface similar to SVN-trac
- Currently we have 4 of these tests implemented; additions welcome!

Score Function: history view

| Revision | Age, days | Author | Earliest SVN revision files equal to | Commit message |
|-----------------------------|-----------|--------|--------------------------------------|--|
| [T] [43377] | 0 | sergey | [42738] | Dummy commit to trigger sfxn tests diff. |
| [T] [43374] | 0 | sergey | [42741] | Dummy commit to trigger sfxn tests diff. |
| [T] [42776] | 28 | sergey | [42738] | Dummy commit to trigger sfxn tests diff. |
| [T] [42761] | 29 | sergey | [42741] | Dummy commit to trigger sfxn tests diff. |
| [T] [42746] | 30 | sergey | [42738] | Dummy commit to trigger sfxn tests diff. |

Score Function: history view

| Revision | Age, days | Author | Earliest SVN revision files equal to | Commit message |
|-----------------------------|-----------|--------|--------------------------------------|--|
| [T] [43377] | 0 | sergey | [42738] | Dummy commit to trigger sfxn tests diff. |
| [T] [43374] | 0 | sergey | [42741] | Dummy commit to trigger sfxn tests diff. |
| [T] [42776] | 28 | sergey | [42738] | Dummy commit to trigger sfxn tests diff. |
| [T] [42761] | 29 | sergey | [42741] | Dummy commit to trigger sfxn tests diff. |
| [T] [42746] | 30 | sergey | [42738] | Dummy commit to trigger sfxn tests diff. |

revision **42741**

revision **42738**

Testing server automatically determines if a freshly committed ScoreFunction equals any previous version, and if so assigns the previous revision number

Benefits of SF fingerprint test

Benefits of SF fingerprint test

- The burden of keeping track of SF moved from developers to TestingServer.

Benefits of SF fingerprint test

- The burden of keeping track of SF moved from developers to TestingServer.
- In publications, you can precisely specify SF version used. Releases could do that as well.

Benefits of SF fingerprint test

- The burden of keeping track of SF moved from developers to TestingServer.
- In publications, you can precisely specify SF version used. Releases could do that as well.
- History-view allows us to keep track of changes in SF: who/when/what



Score Function Fingerprint

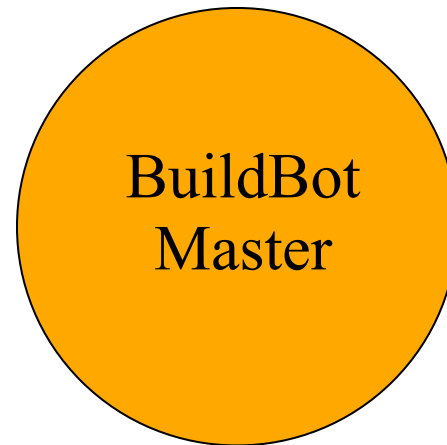


BuildBot server
tracking more builds - a lot of them!

<http://rosettatests.graylab.jhu.edu/buildbot>

BuildBot server

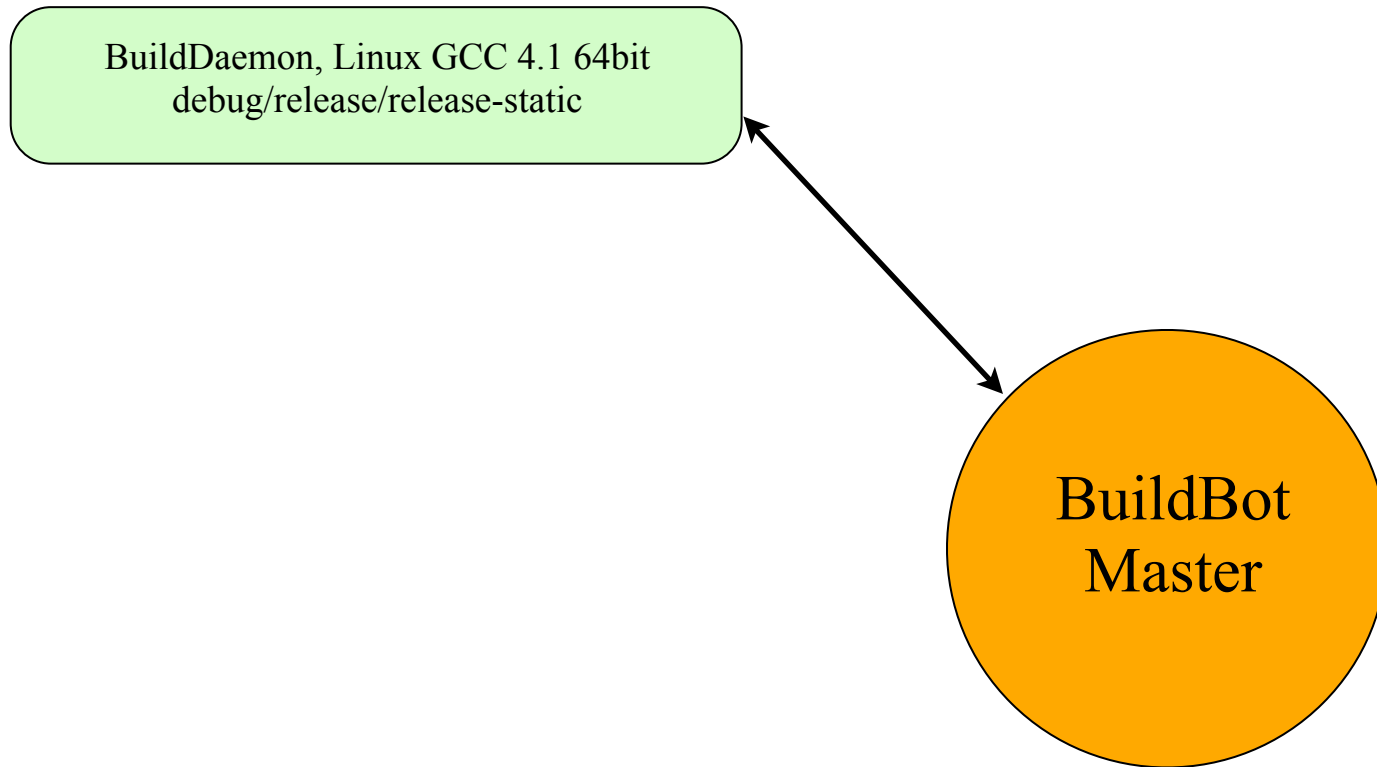
tracking more builds - a lot of them!



<http://rosettatests.graylab.jhu.edu/buildbot>

BuildBot server

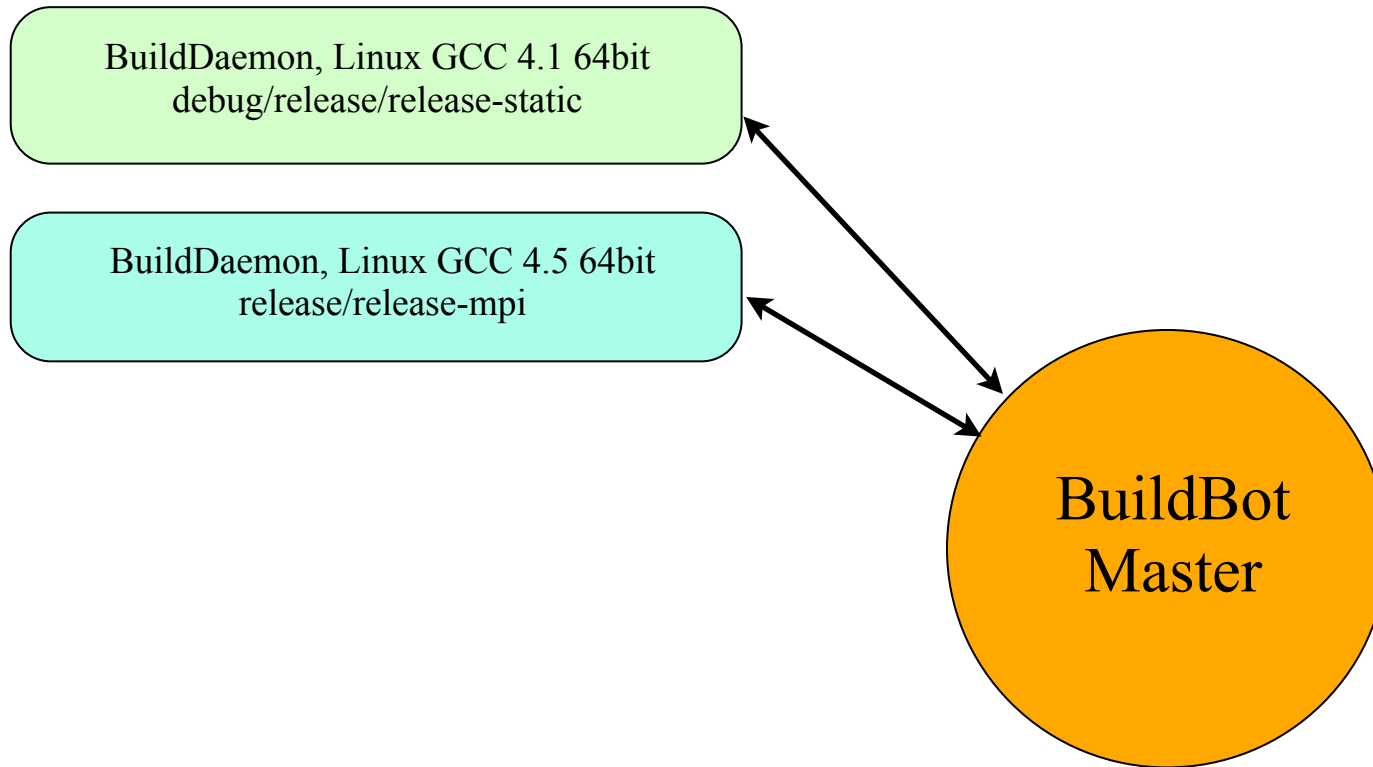
tracking more builds - a lot of them!



<http://rosettatests.graylab.jhu.edu/buildbot>

BuildBot server

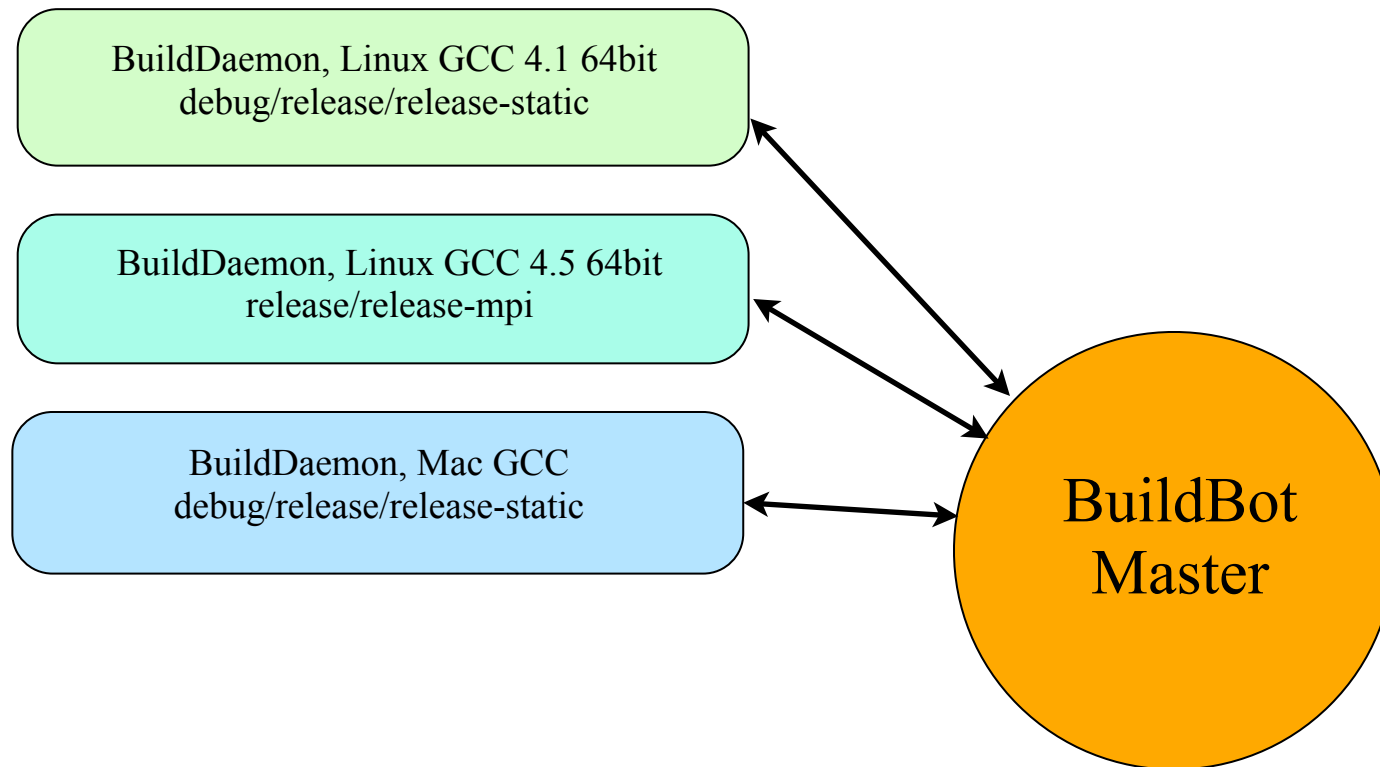
tracking more builds - a lot of them!



<http://rosettatests.graylab.jhu.edu/buildbot>

BuildBot server

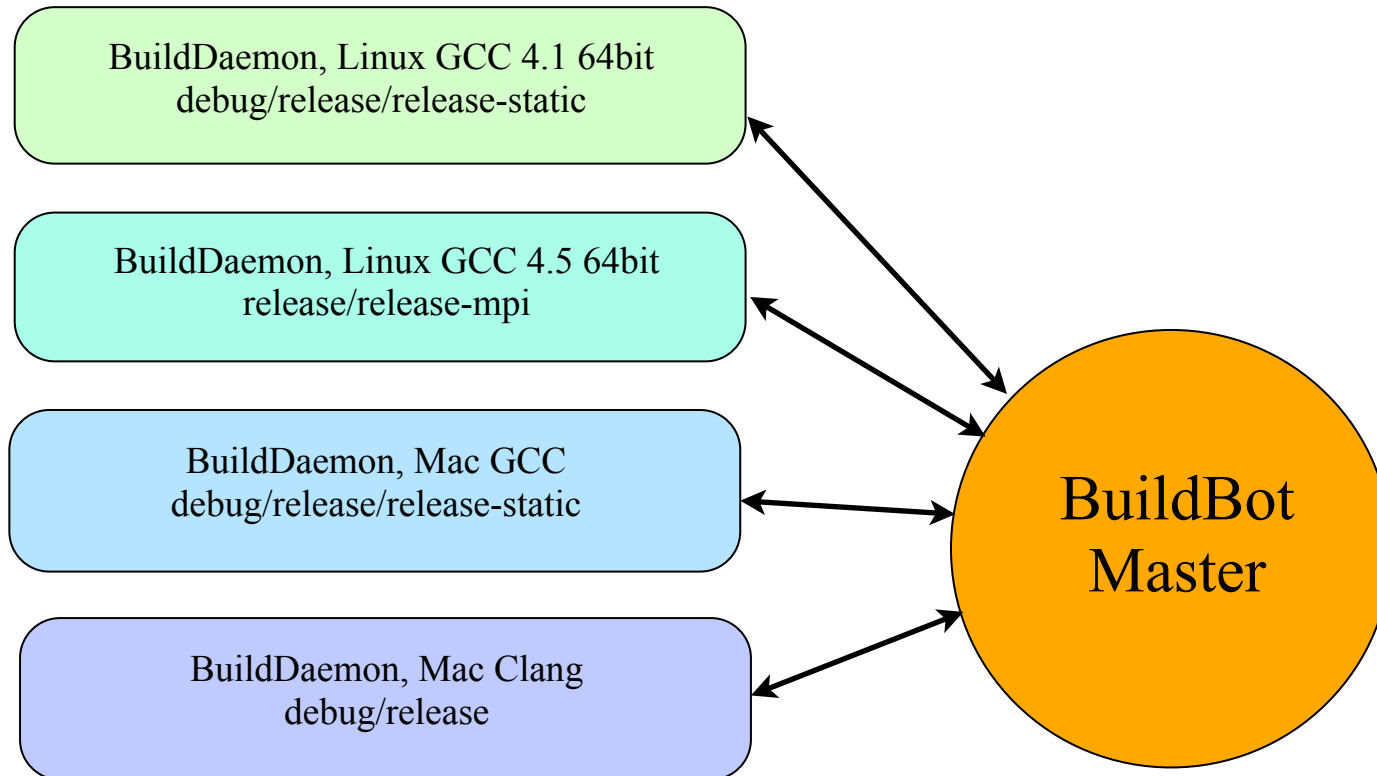
tracking more builds - a lot of them!



<http://rosettatests.graylab.jhu.edu/buildbot>

BuildBot server

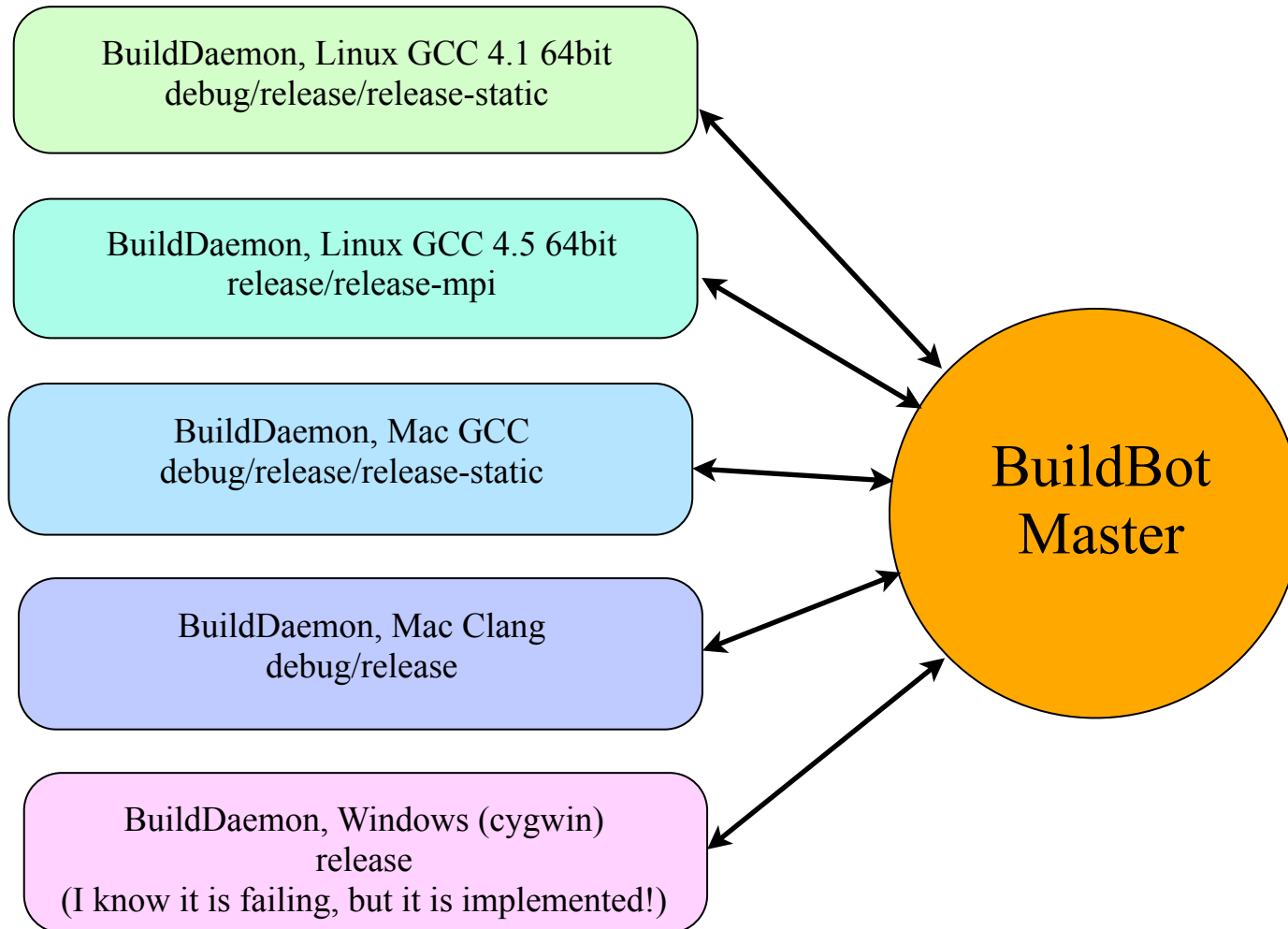
tracking more builds - a lot of them!



<http://rosettatests.graylab.jhu.edu/buildbot>

BuildBot server

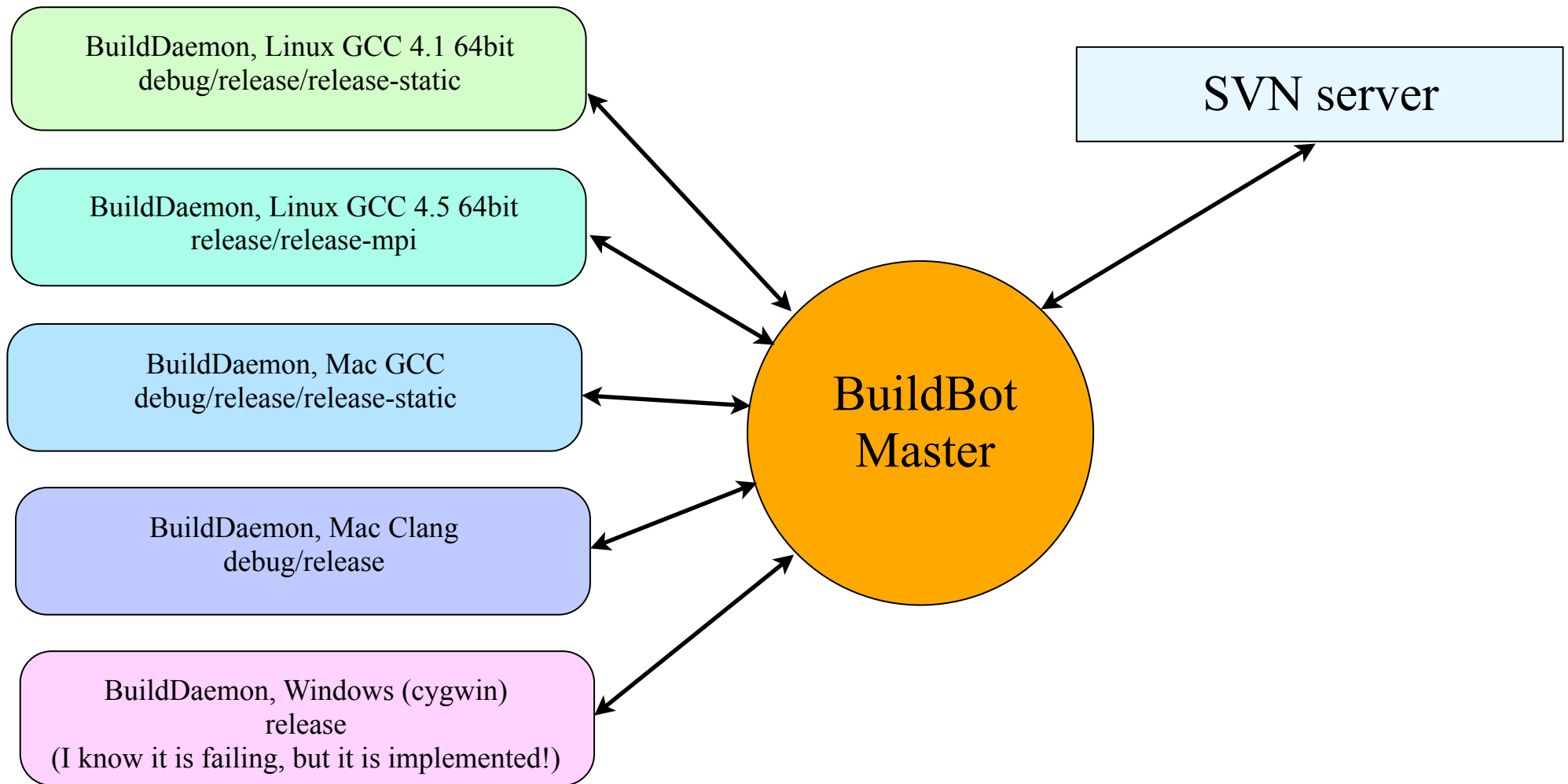
tracking more builds - a lot of them!



<http://rosettatests.graylab.jhu.edu/buildbot>

BuildBot server

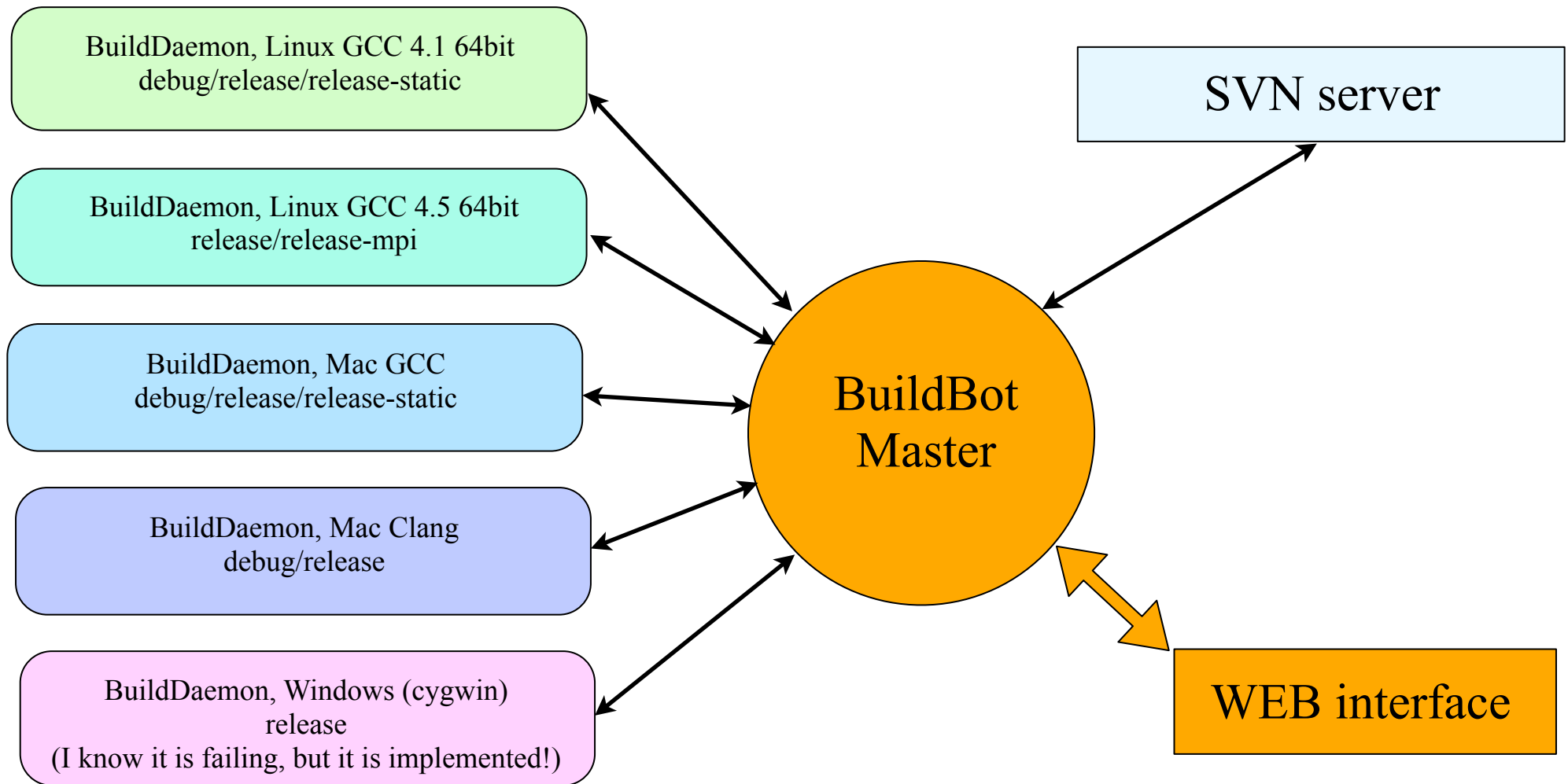
tracking more builds - a lot of them!



<http://rosettatests.graylab.jhu.edu/buildbot>

BuildBot server

tracking more builds - a lot of them!



<http://rosettatests.graylab.jhu.edu/buildbot>

| | | | | | | | | | | | | |
|---|----------|---|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------------|
| Categories: Linux Linux.MPI Mac Mac.Clang Windows | | Legend: Passed Failed Failed Again Running Exception Offline No data | | | | | | | | | | Personalized for... |
| | | | | | | | | | | | | Go |
| | | Linux | | | | Linux.MPI | Mac | | | Mac.Clang | | Windows |
| | | | | | | | | | | | | |
| 43364 | cmiles | | | | | | | | | | | |
| Small update to WeightedReservoirSampler. Ignore non-positive weighted items. | | | | | | | | | | | | |
| 43360 | cmiles | | | | | | | | | | | |
| Library for weighted reservoir sampling and unit tests | | | | | | | | | | | | |
| 43359 | smlewis | | | | | | | | | | | |
| adding a comment to AngleConstraint; it appears to take angles in radians; also updating constraint documentation. Why was this not documented before?? No test changes expected. | | | | | | | | | | | | |
| 43358 | momeara | | | | | | | | | | | |
| Minor bug fixes to features cluster scientific benchmark submission. | | | | | | | | | | | | |
| 43357 | weltzner | | | | | | | | | | | |
| Adding docking prepack to mainpage.dox | | | | | | | | | | | | |
| 43356 | sergey | | | | | | | | | | | |
| Docking docs update. | | | | | | | | | | | | |

Thanks:

Matthew O'Meara and Samuel DeLuca
for setting up and supporting BuildBot
daemons on their sites!

Benefits of BuildBot

Benefits of BuildBot

- We can track a lot of builds - 11 now! And it's really easy to add others.

Benefits of BuildBot

- We can track a lot of builds - 11 now! And it's really easy to add others.
- All builds are incremental - so it is *super fast* even on average machines.

Benefits of BuildBot

- We can track a lot of builds - 11 now! And it's really easy to add others.
- All builds are incremental - so it is *super fast* even on average machines.
- Decentralized system, failure tolerant.

Benefits of BuildBot

- We can track a lot of builds - 11 now! And it's really easy to add others.
- All builds are incremental - so it is *super fast* even on average machines.
- Decentralized system, failure tolerant.
- We tests builds on *exactly* the same systems that run your code.

Benefits of BuildBot

- We can track a lot of builds - 11 now! And it's really easy to add others.
- All builds are incremental - so it is *super fast* even on average machines.
- Decentralized system, failure tolerant.
- We tests builds on *exactly* the same systems that run your code.
- Many machines (6 now) are involved, so it's fast!

Benefits of BuildBot

- We can track a lot of builds - 11 now! And it's really easy to add others.
- All builds are incremental - so it is *super fast* even on average machines.
- Decentralized system, failure tolerant.
- We tests builds on *exactly* the same systems that run your code.
- Many machines (6 now) are involved, so it's fast!
- Builds are non-sequential, we can test revision R and R+1 simultaneously!

Benefits of BuildBot

- We can track a lot of builds - 11 now! And it's really easy to add others.
- All builds are incremental - so it is *super fast* even on average machines.
- Decentralized system, failure tolerant.
- We tests builds on *exactly* the same systems that run your code.
- Many machines (6 now) are involved, so it's fast!
- Builds are non-sequential, we can test revision R and R+1 simultaneously!

Benefits of BuildBot

- We can track a lot of builds - 11 now! And it's really easy to add others.
- All builds are incremental - so it is *super fast* even on average machines.
- Decentralized system, failure tolerant.
- We tests builds on *exactly* the same systems that run your code.
- Many machines (6 now) are involved, so it's fast!
- Builds are non-sequential, we can test revision R and R+1 simultaneously!

Some crazy features to be aware of:

Benefits of BuildBot

- We can track a lot of builds - 11 now! And it's really easy to add others.
- All builds are incremental - so it is *super fast* even on average machines.
- Decentralized system, failure tolerant.
- We tests builds on *exactly* the same systems that run your code.
- Many machines (6 now) are involved, so it's fast!
- Builds are non-sequential, we can test revision R and R+1 simultaneously!

Some crazy features to be aware of:

- You can see current and past console outputs of all builds as they go on the web!

Benefits of BuildBot

- We can track a lot of builds - 11 now! And it's really easy to add others.
- All builds are incremental - so it is *super fast* even on average machines.
- Decentralized system, failure tolerant.
- We tests builds on *exactly* the same systems that run your code.
- Many machines (6 now) are involved, so it's fast!
- Builds are non-sequential, we can test revision R and R+1 simultaneously!

Some crazy features to be aware of:

- You can see current and past console outputs of all builds as they go on the web!
- You can request (re)build for a particular revision using web interface!

Future plans for BuildBot

Future plans for BuildBot

- Get a dedicated hi-end machine that will run numerous virtual machines. This would allow to expand the range of tested architectures.

Future plans for BuildBot

- Get a dedicated hi-end machine that will run numerous virtual machines. This would allow to expand the range of tested architectures.
- “Bleeding-edge” builds on latest versions of GCC-4.6 and Clang-3.0.

Future plans for BuildBot

- Get a dedicated hi-end machine that will run numerous virtual machines. This would allow to expand the range of tested architectures.
- “Bleeding-edge” builds on latest versions of GCC-4.6 and Clang-3.0.
- Run unit tests on various platforms. Hopefully we can have native Windows build!

Future plans for BuildBot

- Get a dedicated hi-end machine that will run numerous virtual machines. This would allow to expand the range of tested architectures.
- “Bleeding-edge” builds on latest versions of GCC-4.6 and Clang-3.0.
- Run unit tests on various platforms. Hopefully we can have native Windows build!
- Valgrind runs.

Future plans for BuildBot

- Get a dedicated hi-end machine that will run numerous virtual machines. This would allow to expand the range of tested architectures.
- “Bleeding-edge” builds on latest versions of GCC-4.6 and Clang-3.0.
- Run unit tests on various platforms. Hopefully we can have native Windows build!
- Valgrind runs.
- PyRosetta builds.





SVN Statistics

SVN Statistics!

SVN Statistics!

Right now mini source code is ~ **1,426,000** lines long

SVN Statistics!

Right now mini source code is ~ **1,426,000** lines long
[Last year was at ~1,163,000 lines, so it grew ~23%]

SVN Statistics!

Right now mini source code is ~ **1,426,000** lines long
[Last year was at ~1,163,000 lines, so it grew ~23%]

Since last year we have **2,659** revisions committed to
mini trunk, thats **~7.3** rev/day

SVN Statistics!

Right now mini source code is $\sim 1,426,000$ lines long
[Last year was at $\sim 1,163,000$ lines, so it grew $\sim 23\%$]

Since last year we have **2,659** revisions committed to
mini trunk, that's ~ 7.3 rev/day
[Last year: 2,187 ~ 6 rev/day]

Student/postdoc with highest rate of broken builds: **Elizabeth H.Kellogg**

Time frame: Sort by: Show only users who committed at least: revisions

| User | Revisions committed | Builds broken (%) | Unit tests broken (%) | Integration tests changed (%) |
|-----------------|---------------------|-------------------|-----------------------|-------------------------------|
| ekellogg | 30 | 4 (13%) | 0 (0%) | 3 (10%) |
| possu | 25 | 3 (12%) | 0 (0%) | 2 (8%) |
| johnk | 26 | 3 (11%) | 0 (0%) | 5 (19%) |
| Nikolas | 20 | 2 (10%) | 0 (0%) | 2 (10%) |
| chrisk | 32 | 3 (9%) | 0 (0%) | 11 (34%) |
| dgront | 54 | 5 (9%) | 1 (1%) | 19 (35%) |
| olange | 97 | 8 (8%) | 2 (2%) | 41 (42%) |
| tex | 141 | 11 (7%) | 2 (1%) | 29 (20%) |
| delucasl | 40 | 3 (7%) | 0 (0%) | 6 (15%) |
| scombs | 45 | 3 (6%) | 1 (2%) | 16 (35%) |

Student/postdoc with highest rate of broken builds: Elizabeth H.Kellogg

Time frame: last_year Sort by: build Show only users who committed at least: 16 revisions Go!

| User | Revisions committed | Builds broken (%) | Unit tests broken (%) | Integration tests changed (%) |
|----------|---------------------|-------------------|-----------------------|-------------------------------|
| ekellogg | 30 | 4 (13%) | 0 (0%) | 3 (10%) |
| possu | 25 | 3 (12%) | 0 (0%) | 2 (8%) |
| johnk | 26 | 3 (11%) | 0 (0%) | 5 (19%) |
| Nikolas | 20 | 2 (10%) | 0 (0%) | 2 (10%) |
| chrisk | 32 | 3 (9%) | 0 (0%) | 11 (34%) |
| dgront | 54 | 5 (9%) | 1 (1%) | 19 (35%) |
| olange | 97 | 8 (8%) | 2 (2%) | 41 (42%) |
| tex | 141 | 11 (7%) | 0 (0%) | 0 (0%) |
| delucasl | 40 | 3 (7%) | 0 (0%) | 0 (0%) |
| scombs | 45 | 3 (6%) | 1 (2%) | 16 (35%) |

Elizabeth H.Kellogg

Possu Huang

John Karanicolas

Nikolas Sgourakis

Last year highest rate was: 18%

Student/postdoc with highest rate of broken Unit tests: **Gordon Lemmon**

Time frame: Sort by: Show only users who committed at least: revisions [Go!](#)

| User | Revisions committed | Builds broken (%) | Unit tests broken (%) | Integration tests changed (%) |
|-----------------|---------------------|-------------------|-----------------------|-------------------------------|
| glemmon | 38 | 1 (2%) | 2 (5%) | 15 (39%) |
| kenjung | 35 | 2 (5%) | 1 (2%) | 9 (25%) |
| dimaio | 78 | 2 (2%) | 2 (2%) | 28 (35%) |
| scombs | 45 | 3 (6%) | 1 (2%) | 16 (35%) |
| leaverfa | 95 | 4 (4%) | 2 (2%) | 32 (33%) |
| olange | 97 | 8 (8%) | 2 (2%) | 41 (42%) |
| dgront | 54 | 5 (9%) | 1 (1%) | 19 (35%) |
| momeara | 166 | 8 (4%) | 3 (1%) | 59 (35%) |
| tex | 141 | 11 (7%) | 2 (1%) | 29 (20%) |
| cmiles | 202 | 6 (2%) | 2 (0%) | 27 (13%) |

Student/postdoc with highest rate of broken Unit tests: **Gordon Lemmon**

Time frame: Sort by: Show only users who committed at least: revisions

| User | Revisions committed | Builds broken (%) | Unit tests broken (%) | Integration tests broken (%) |
|----------|---------------------|-------------------|-----------------------|------------------------------|
| glemmon | 38 | 1 (2%) | 2 (5%) | 15 (39%) |
| kenjung | 35 | 2 (5%) | 1 (2%) | 1 (25%) |
| dimaio | 78 | 2 (2%) | 2 (2%) | 28 (35%) |
| scombs | 45 | 3 (6%) | 1 (2%) | 1 (35%) |
| leaverfa | 95 | 4 (4%) | 2 (2%) | 32 (33%) |
| olange | 97 | 8 (8%) | 2 (2%) | 41 (42%) |
| dgront | 54 | 5 (9%) | 1 (1%) | 19 (35%) |
| momeara | 16 | 1 (6%) | 1 (6%) | 59 (35%) |
| tex | 141 | 11 (7%) | 2 (1%) | 29 (20%) |
| cmiles | 202 | 6 (2%) | 2 (0%) | 27 (13%) |

Gordon Lemmon

Kenneth Jung

Frank Dimaio

Steven Combs

Last year highest rate was: **11%**

Student/postdoc with highest rate of Integration tests change: **Monica Berrondo**

Time frame: Sort by: Show only users who committed at least: revisions

| User | Revisions committed | Builds broken (%) | Unit tests broken (%) | Integration tests changed (%) |
|---------|---------------------|-------------------|-----------------------|-------------------------------|
| monica | 16 | 1 (6%) | 0 (0%) | 9 (56%) |
| andre | 26 | 1 (3%) | 0 (0%) | 11 (42%) |
| olange | 97 | 8 (8%) | 2 (2%) | 41 (42%) |
| glemmon | 38 | 1 (2%) | 2 (5%) | 15 (39%) |
| csmith | 21 | 0 (0%) | 0 (0%) | 8 (38%) |
| barak | 24 | 1 (4%) | 0 (0%) | 9 (37%) |
| dimaio | 78 | 2 (2%) | 2 (2%) | 28 (35%) |
| scombs | 45 | 3 (6%) | 1 (2%) | 16 (35%) |
| momeara | 166 | 8 (4%) | 3 (1%) | 59 (35%) |
| dgront | 54 | 5 (9%) | 1 (1%) | 19 (35%) |

SLast year highest rate was the same: **56%**

ests change: **Monica Berrondo**

Time frame: Sort by: Show only users who committed at least: revisions

| User | Revisions committed | Builds broken (%) | Unit tests broken (%) | Integration tests broken (%) |
|---------|---------------------|-------------------|-----------------------|------------------------------|
| monica | 16 | 1 (6%) | 0 (0%) | 9 (56%) |
| andre | 26 | 1 (3%) | 0 (0%) | 11 (42%) |
| olange | 97 | 8 (8%) | 2 (2%) | 41 (42%) |
| glemmon | 38 | 1 (2%) | 0 (0%) | 15 (39%) |
| csmith | 21 | 0 (0%) | 0 (0%) | 8 (38%) |
| barak | 24 | 1 (4%) | 0 (0%) | 9 (37%) |
| dimaio | 78 | 2 (2%) | 2 (2%) | 28 (35%) |
| scombs | 45 | 3 (6%) | 1 (2%) | 16 (35%) |
| momeara | 166 | 8 (4%) | 3 (1%) | 59 (35%) |
| dgront | 54 | 5 (9%) | 1 (1%) | 19 (35%) |

Monica Berrondo

Ingemar Andre

Oliver Lange

Gordon Lemmon

Colin A. Smith

Barak Raveh

Student/postdoc who committed most revisions: Mike Tyka

Time frame: Sort by: Show only users who committed at least: revisions [Go!](#)

| User | Revisions committed | Builds broken (%) | Unit tests broken (%) | Integration tests changed (%) |
|----------|---------------------|-------------------|-----------------------|-------------------------------|
| sergey | 279 | 6 (2%) | 1 (0%) | 27 (9%) |
| mtyka | 260 | 12 (4%) | 2 (0%) | 49 (18%) |
| cmiles | 202 | 6 (2%) | 2 (0%) | 27 (13%) |
| sarel | 192 | 7 (3%) | 1 (0%) | 60 (31%) |
| momeara | 166 | 8 (4%) | 3 (1%) | 59 (35%) |
| tex | 141 | 11 (7%) | 2 (1%) | 29 (20%) |
| sheffler | 98 | 0 (0%) | 0 (0%) | 2 (2%) |
| olange | 97 | 8 (8%) | 2 (2%) | 41 (42%) |
| leaverfa | 95 | 4 (4%) | 2 (2%) | 32 (33%) |
| dimaio | 78 | 2 (2%) | 2 (2%) | 28 (35%) |
| smlewis | 77 | 0 (0%) | 0 (0%) | 20 (25%) |

Student/postd

ke Tyka

Last year highest number was: 204

Time frame: Sort by:

| User | Revisions com | | | |
|----------|---------------|---------|--------|----------|
| sergey | 279 | 6 (2%) | 1 (0%) | 27 (9%) |
| mtyka | 260 | 12 (4%) | 2 (0%) | 49 (18%) |
| cmiles | 202 | 6 (2%) | 2 (0%) | 27 (13%) |
| sarel | 192 | 7 (3%) | 1 (0%) | 31 (16%) |
| momeara | 166 | 8 (4%) | 3 (1%) | 59 (35%) |
| tex | 141 | 11 (7%) | 2 (1%) | 20 (14%) |
| sheffler | 98 | 0 (0%) | 0 (0%) | 2 (2%) |
| olange | 97 | 8 (8%) | 2 (2%) | 41 (42%) |
| leaverfa | 95 | 4 (4%) | 2 (2%) | 32 (34%) |
| dimaio | 78 | 2 (2%) | 2 (2%) | 28 (36%) |
| smlewis | 77 | 0 (0%) | 0 (0%) | 20 (26%) |

Mike Tyka

Christopher Miles

Sarel Fleishman

Matthew O'Meara

James Thompson

William Sheffler

Oliver Lange

Andrew Leaver-Fay

Frank Dimaio

Steven Lewis

Next priorities

Next priorities

1. More score function tests.

Next priorities

1. More score function tests.
2. New dedicated machine for BuildBot builds (?)

Next priorities

1. More score function tests.
2. New dedicated machine for BuildBot builds (?)
3. More BuildBot builds on different platforms.

Next priorities

1. More score function tests.
2. New dedicated machine for BuildBot builds (?)
3. More BuildBot builds on different platforms.
4. New faster Test server for revision-by-revision daemon.

Special Thanks to:

Steven Lewis

Matthew O'Meara

Vladimir Yarov-Yarovoy

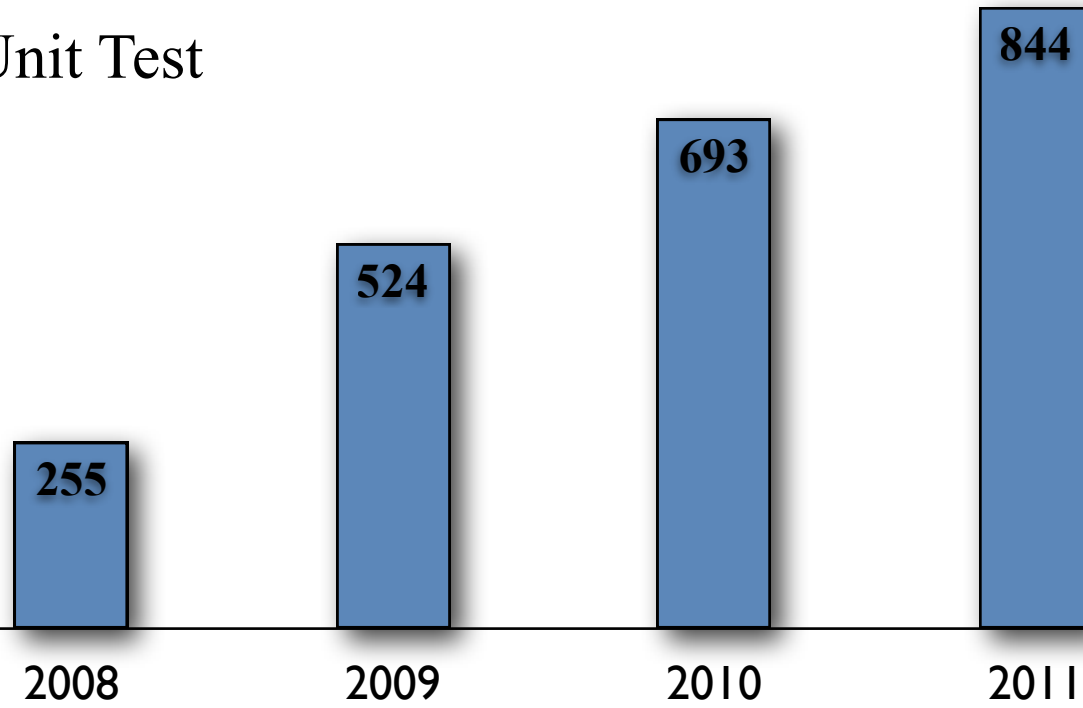
and to everyone else who helped develop and
maintain these tests, feedback, and bug
reports.

Thank you!

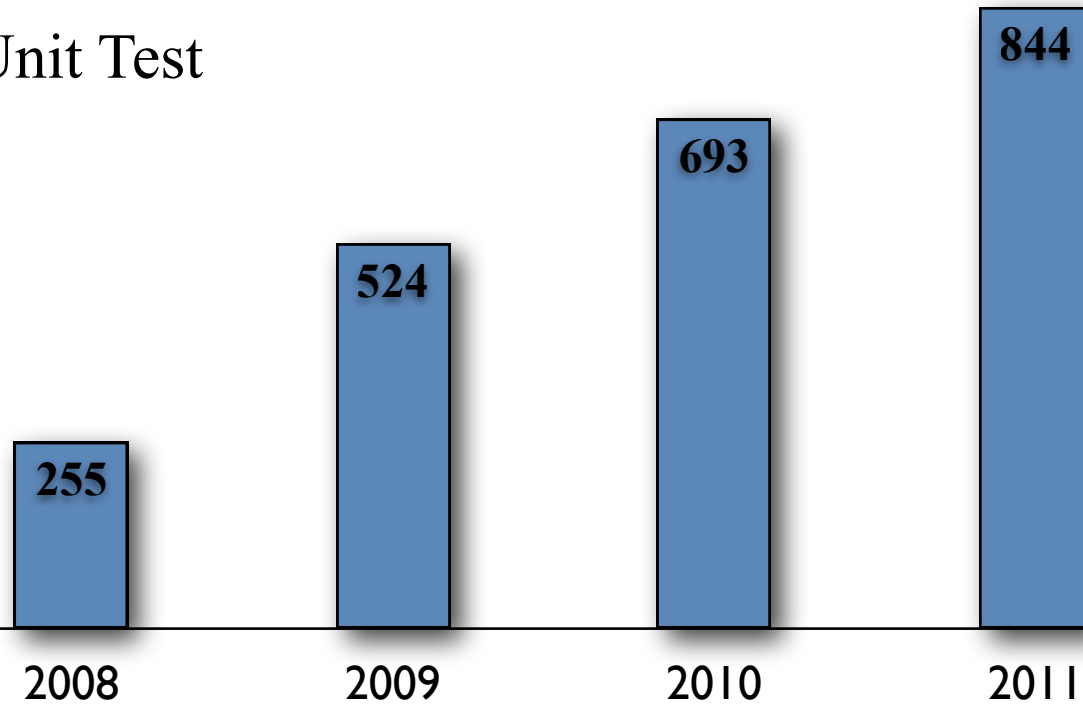
List of currently implemented builds:

- Scientific Linux-64 GCC 4.1.2: pilot apps debug/release/release-static
- CentOS Linux-64 GCC 4.5.1 pilot apps release/release-mpi
- Mac GCC 4.2 pilot apps debug/release/release-static
- Mac Clang debug/release
- Windows (cygwin) (I know it is failing, but it is implemented!)

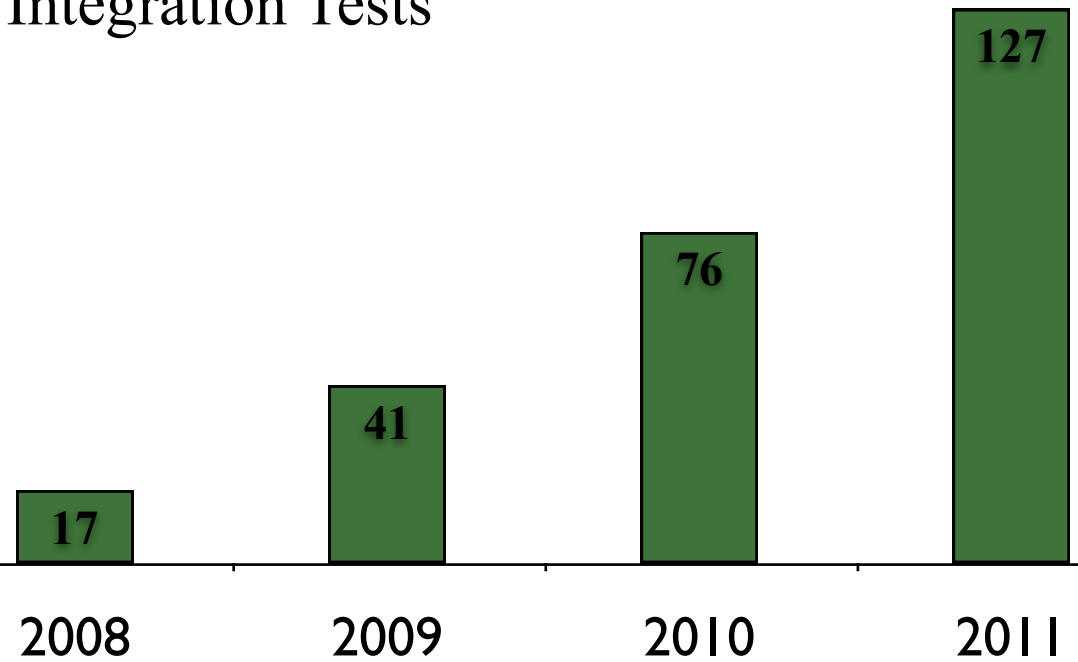
Unit Test



Unit Test

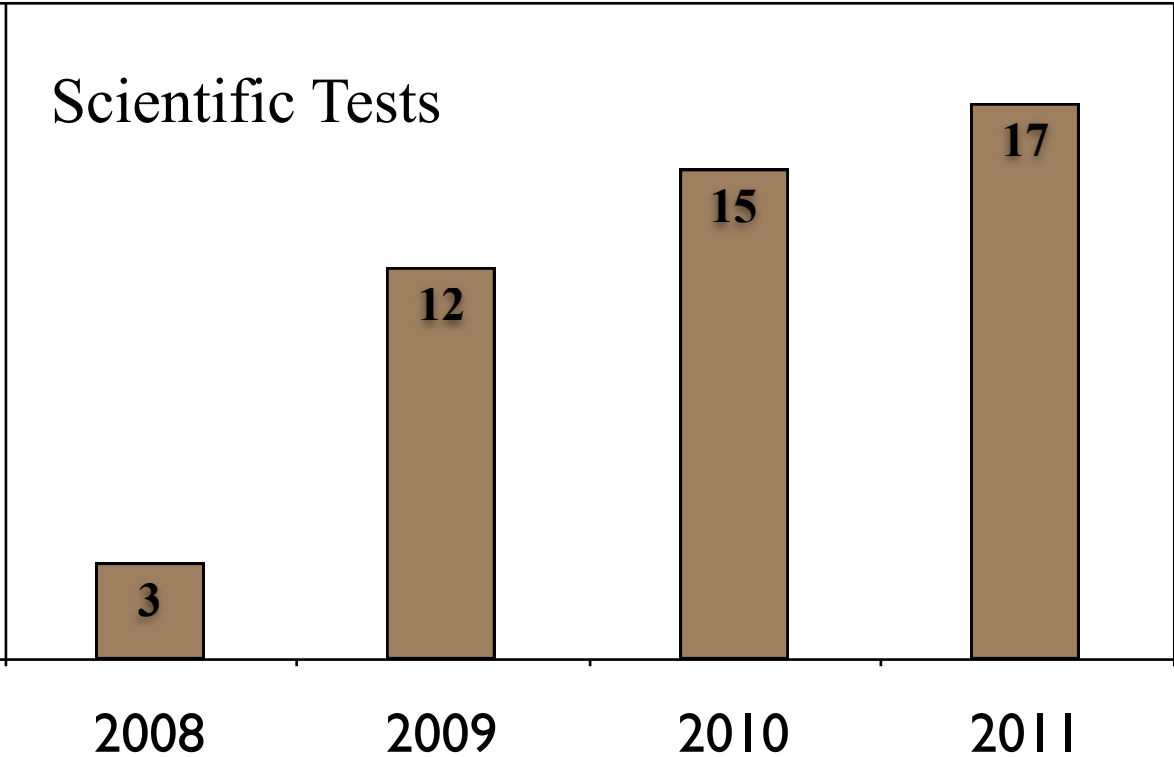


Integration Tests



| | 2010 | 2011 |
|----------------------------------|------|------|
| Profile Tests | 9 | 11 |
| Performance Tests | 9 | 8 |
| Score Function fingerprint Tests | - | 3 |

| | 2010 | 2011 |
|----------------------------------|------|------|
| Profile Tests | 9 | 11 |
| Performance Tests | 9 | 8 |
| Score Function fingerprint Tests | - | 3 |



[2010] User with highest rate of broken builds: **Oliver Lange**

Time frame: Sort by:

| User | Revisions committed | Builds broken (%) | Unit tests broken (%) | Integration tests changed (%) |
|-----------------|---------------------|-------------------|-----------------------|-------------------------------|
| olange | 64 | 12 (18%) | 0 (0%) | 22 (34%) |
| possu | 36 | 5 (13%) | 0 (0%) | 7 (19%) |
| aroop | 38 | 5 (13%) | 0 (0%) | 12 (31%) |
| dgront | 85 | 11 (12%) | 1 (1%) | 27 (31%) |
| ekellogg | 74 | 8 (10%) | 0 (0%) | 5 (6%) |
| barak | 20 | 2 (10%) | 0 (0%) | 7 (35%) |
| sid | 42 | 4 (9%) | 2 (4%) | 11 (26%) |
| renfrew | 21 | 2 (9%) | 1 (4%) | 8 (38%) |
| sheffler | 43 | 4 (9%) | 0 (0%) | 2 (4%) |
| rhiju | 44 | 4 (9%) | 4 (9%) | 10 (22%) |
| andre | 49 | 4 (8%) | 0 (0%) | 11 (22%) |
| glemmon | 25 | 2 (8%) | 0 (0%) | 14 (56%) |

List of currently implemented scientific tests:

1. abinitio
2. dna_interface_design
3. docking
4. ligand_docking
5. loop
6. membrane
7. monomer_ddg
8. multi_residue_ligand_docking
9. relax
- 10.rna_denovo
- 11.rna_design
- 12.sequence_recovery
- 13.detailed_balance
- 14.enzdes_benchmark
- 15.Rotamer Recovery