Server Side Scripting in Java
SECR 2007

Sergey Salishev

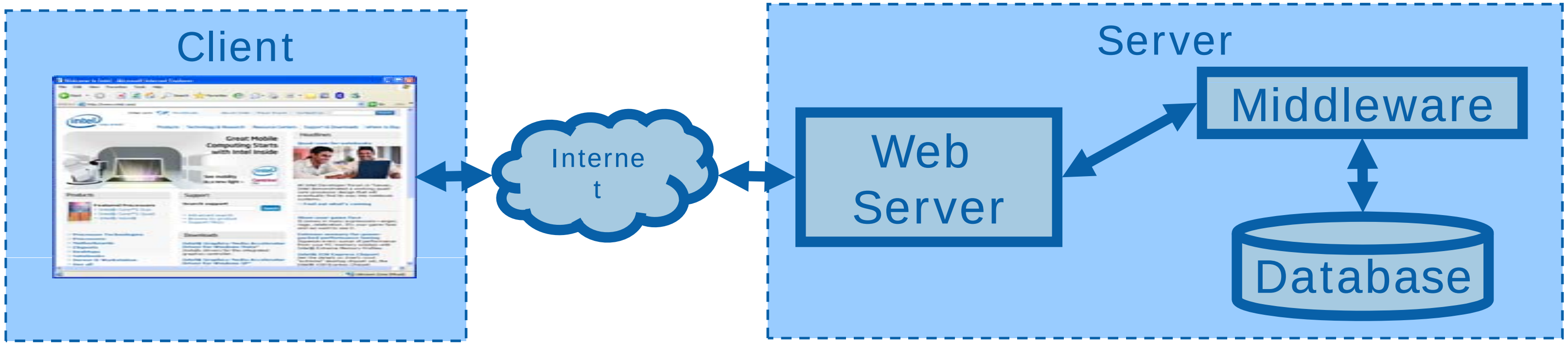
sergey.i.salishev@intel.com

Agenda

- What is Server side scripting?
- Motivation for Server side scripting in Java
- Benchmark
  - Architecture
  - Choosing the Workload
  - Calculating the the Score
- Native Scripting
- JSR223 experience
- Java based Scripting
- Performance Comparison
- Java CPU cycle distribution
- Native CPU cycle distribution
- Conclusions
- Q&A

What is Server side scripting?

Typical 3-tier web application



The common Opens Source Server platforms are:

- LAMP (Linux, Apache, MySQL, PHP/Perl/Python)
- J2EE with custom UNIX and database

Middleware implements business logics and interacting with client and database

- Higher volatility and support requirements compared to other software

Middleware for these platforms is written in

- script (PHP/Perl/Python/Ruby)
- Java

Motivation for Server side scripting in Java

- J2EE has now fully Open Source implementations
- Eclipse* greatly simplifies development of Web Applications in Java
- Large number of existing scripts
- Java is good for components and complex solutions
- Scripts are good for gluing it up with the web interface
- A number of compatible Open Source Script engines in Java
- Both platforms provide the full stack for web development

Is it possible to merge both worlds on a common Java platform?

** Eclipse is a traidmark of Eclipse Foundation, Inc.*

Benchmark

How to assess the feasibility of replacing native Script engines with Java solutions?

- **Is it possible? Is it difficult?**
- **How does it perform compared to native?**

Need an universal benchmark based on real life usage scenario

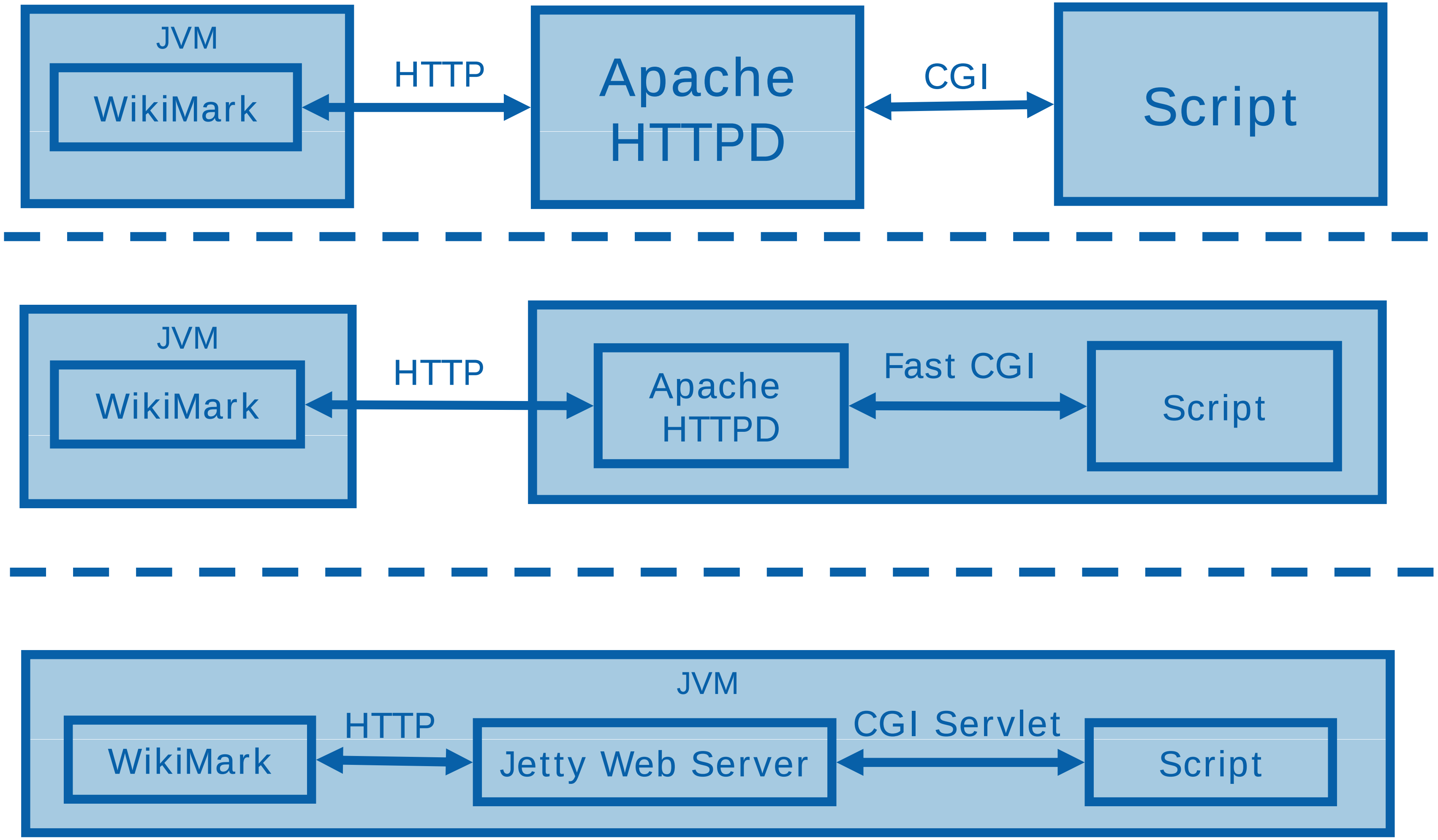
Typical scripting script usages:

- **Wiki (Wikipedia)**
- Forum
- CMS (Content Management System)
- E-Shop

All Wiki have similar syntax. Easy to develop script neutral benchmark.

Need to choice a realistic workload

WikiMark architecture



Choosing the workload

Need to approximate the page sources size distribution

$$S_{source} = C\ (S_{page} - S_{template})$$

C – markup size diff approximated to 1

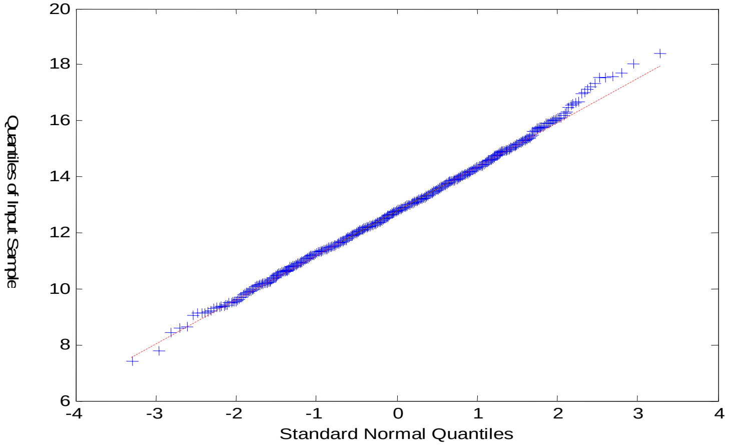
From en.wikipedia.org 1000 random pages were taken

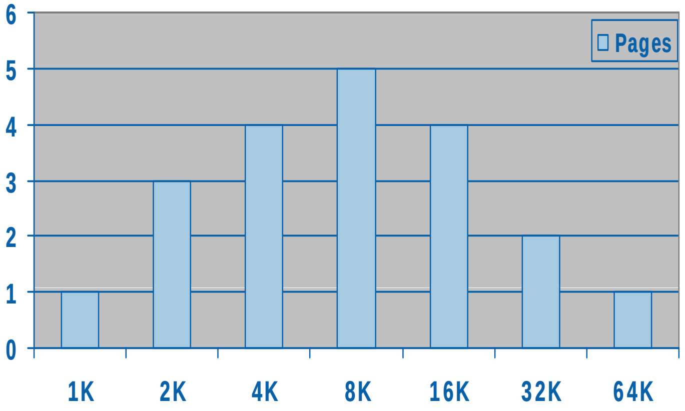
It's 0.05% of 2,053,424 total page population

The $\log_2 S_{source}$ is normally distributed as ***N****(12.7939, 1.59413)*

95% page source sizes are between 1K and 64K

Use the fixed page size distribution to make reproducible results





Calculating the score

1. The random set of cross-referenced pages with a complex formatting is generated with predefined size distribution.
2. All page are requested once using *k* threads from 1 to 2P.
3. Throughput $T_{i,k}$ is calculated.
4. The experiment repeated for *N* times.
5. The score is the maximum across the thread counts of the throughputs trimmed mean.

$$S = max_t \mathbf{M}[T_{t,i}]$$

Native Scripting

PHP 5.2.4

- Only Fast CGI. No problems.

Ruby 1.8.6

- CGI. No problems.
- Fast CGI. **mod_ruby 1.2.6**
  - Needed to lower the default security level.

Python 2.5.1

- CGI. No problems.
- Fast CGI. **mod_python 3.3.1**
  - cgi module is broken. Needed to modify the script to workaround.

JSR 223 experience

Tried to use JSR 223 API for emulating CGI

Problems in API:

- No generic way to destroy the ScriptEngine
- Can’t set the Environment and cwd
- Can’t pass the script file name directly to the script engine

Problems in pluggable engines:

- No engines handle ARGV attribute
- No engines handle overriding standard IO streams
- No engines except Quercus marshal Java Map to script type

Java based Scripting

Quercus* 3.1.2

- No problems

JRuby SVN head (4779)

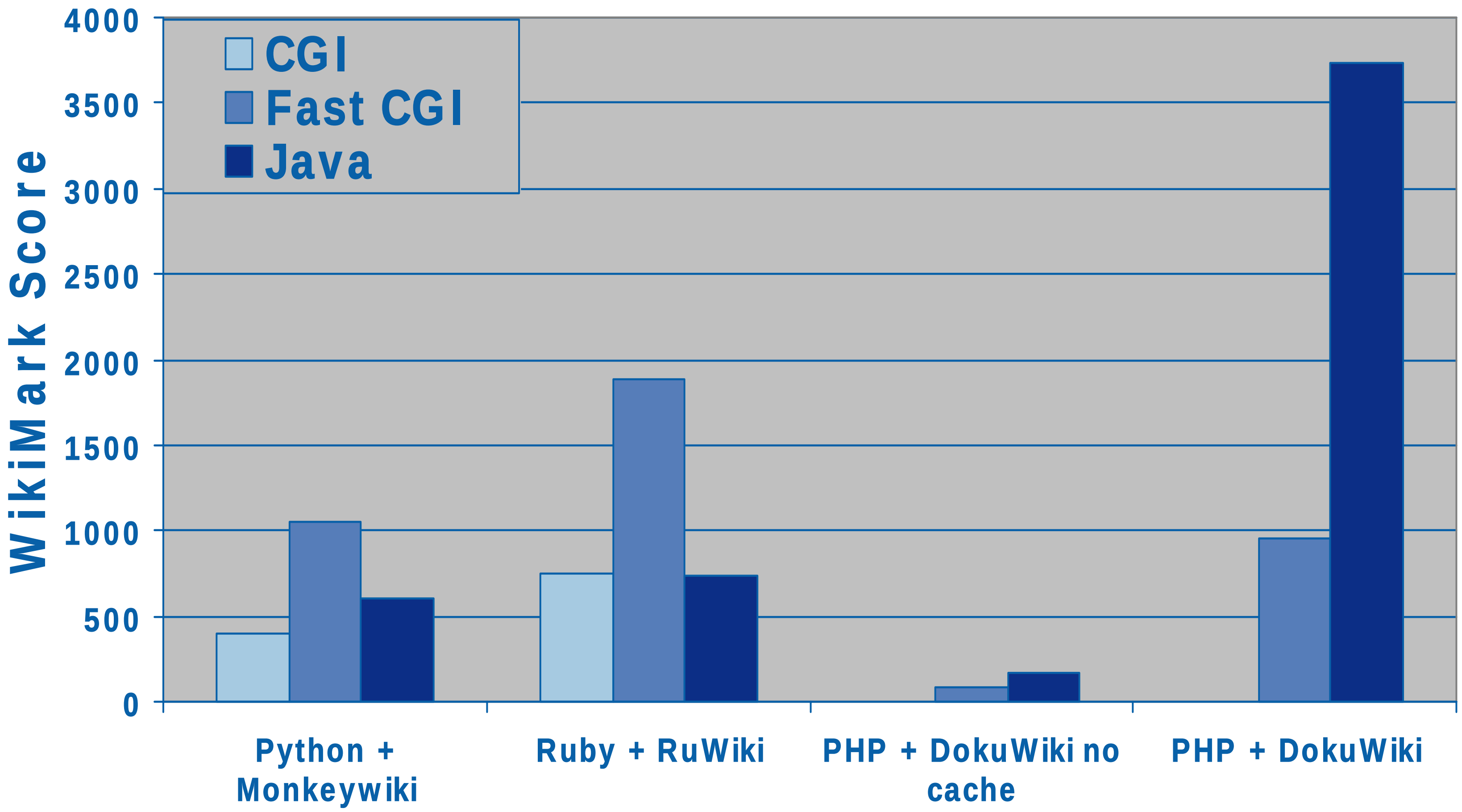
- Problems with IO, strings, regex, thread safety
- No JIT’ed code sharing between runtimes
- Needed to fix some bugs to run the workload

Jython SVN head (3613)

- Severe problems with IO, strings, thread safety.
- Needed to heavily modify the jython code to run the workload

* *Quercus is a trademark of Caucho Technology, Inc.*

Performance Comparison



Java CPU cycle distribution in %

Engine + Wiki	JVM	JIT'ed	Regex	Script Code	Call
Jython + Monkeywiki	4.3	88	43.2	2.7	~8.5
JRuby + RuWiki	6	79	41.3	2.6	~3
Quercus + DokuWiki no cache	3.5	89.4	39.6	7.6	~1.3
Quercus + DokuWiki	8.7	62.2	8.7	5.4	~1.9

Native CPU cycle distribution in %

Engine + Wiki	Engine Lib	Libc	Syscall	Regex
Python + Monkeywiki Fast CGI	85.7	7.8	4.2	42.8
Ruby + RuWiki Fast CGI	74.3	16.4	6.8	32.5
PHP + DokuWiki no cache	92.1	5	2.5	31
PHP + DokuWiki	82.2	8	8.3	0.4

Conclusions

Java based scripting showed adequate performance in all cases

Serious quality problems with Jython and JRuby

JSR223 is incomplete

Need to fix this to promote the Java based scripting

More time in libraries and less time in the scripting code

Optimizing libraries is more beneficial

Q&A