Deployment and Monitoring

Pascal Robert
MacTI

Contents

- Deployment
  - Standard wotaskd/JavaMonitor
  - Wonder's wotaskd/JavaMonitor
  - Alternatives
- Monitoring
  - Nagios
  - JMX

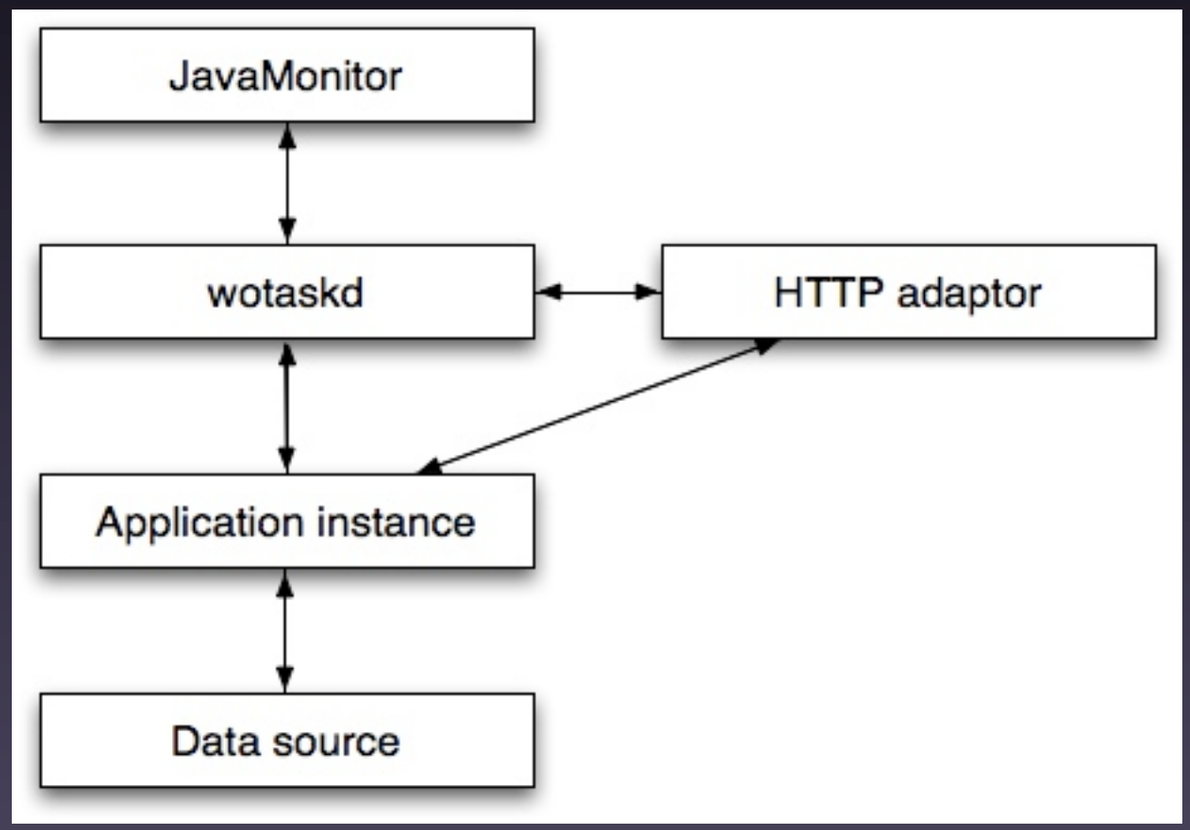
wotaskd/JavaMonitor

- Bundled with WO, as two WO applications
- At least one wotaskd per server needed, one JavaMonitor on one host can talk to multiple wotaskds
- JavaMonitor is just a Web app to manage wotaskd's configuration file (SiteConfig.xml)

wotaskd/JavaMonitor

- wotaskd will communicate with the applications it manage
- Can run multiple wotaskd on the same host, on different ports/IP (for virtual hosting or tests environment; you also need multiple HTTP adaptor/server)
- HTTP Adaptor for your Web server software (Apache, IIS) is also needed, adaptor will talk to wotaskd to fetch the applications available to use

wotaskd/JavaMonitor



wotaskd/JavaMonitor

- By default, it's insecure, especially on servers with public IPs
  - wotaskd TCP port (1085) is open to everyone
  - JavaMonitor password is sent in clear text
  - JavaMonitor can be prone to password dictionary attacks

wotaskd/JavaMonitor

- To make it more secure :
  - Close wotaskd TCP and UDP port at the firewall level, allowed connections only from your JavaMonitor and Web servers hosts.
  - Start JavaMonitor under HTTPS, or...
  - Start JavaMonitor on demand via a SSH tunnel
- Let's see how to secure JavaMonitor...

DEMO

Self-signed certificate on Mac OS X "client"

```
$ sudo cd /etc/apache2

$ sudo openssl genrsa -out leopards.macti.lan.key 2048

$ sudo openssl req -new -key leopards.macti.lan.key -out
leopards.macti.lan.csr

$ sudo openssl x509 -req -days 365 -in leopards.macti.lan.csr -
signkey leopards.macti.lan.key -out leopards.macti.lan.crt
```

wotaskd/JavaMonitor

- The most reported problem with launching apps with wotaskd : file permissions.
- On Mac OS X (10.4 and 10.5), wotaskd and JavaMonitor are running under the *appserver* user and *appserverusr* group.
- Make sure that the user who install the application is a member of the *appserverusr* and *appserveradm* groups.
- Make sure that *appserver* can access your application's directory and startup script.

  /Library/WebObjects/Applications/Myapp.woa/MyApp

wotaskd/JavaMonitor

- If the app doesn't start and no logs are written, it migth be a classpath problem.
    - Start your app by command line with the same arguments as JavaMonitor.
    - Check the output for any errors.
    - By default, the appserver user don't have a valid shell, change it with dscl. Sample :

      dscl . change /users/_appserver UserShell /usr/bin/false /bin/bash
    - Run the app as *appserver*

      su - appserver /Library/WebObjects/App.woa/App

wotaskd/JavaMonitor

- Last step : check your DNS! Make sure you have both forward (name -> IP) and reverse (IP -> name) entries for your host.

  $ host wocommunity.org
  wocommunity.org has address 70.38.69.25

  $ host 70.38.69.25
  25.69.38.70.in-addr.arpa domain name pointer wocommunity.org

- If you don't have a domain name, create one. It's easy to do in Server Admin or Webmin.

DEMO

Deployment tasks

- Build your app with ant/Hudson/other alternatives
- Upload the application in /Library/WebObjects/Applications
- Upload the WSR in /Library/WebServer/Documents/WebObjects
- Restart the application (or add it if it doesn't exist) in JavaMonitor

WOWODC!

DEPLOYMENT DEMO

```
$ scp /Library/WebObjects/Applications/Uber-Application.tar.gz
admin@leopards.macti.lan:

$ scp /Library/WebObjects/Applications/Uber-
WebServerResources.tar.gz admin@leopards.macti.lan:

$ ssh admin@leopards.macti.lan

$ tar zxf Uber-Application.tar.gz

$ mv Uber.woa /Library/WebObjects/Applications/

$ tar zxf Uber-WebServerResources.tar.gz

$ mv Uber.woa /Library/WebServer/Documents/WebObjects/

$ chgrp appserveradm /Library/WebObjects/Applications/Uber.woa/
Uber
```

That's a lot of steps...

Automated deployment

- You can execute SSH tasks from ant with jsch
- So let's add some SSH calls in build.xml

DEMO

wotaskd/JavaMonitor

- wotaskd and JavaMonitor are open source...
- And Project Wonder has customized versions of the tools!
- So how it is different from the standard version?

Wonder's wotaskd/ Monitor

- Wonder's version of JavaMonitor offers direct actions for :
  - Collecting statistics about the instances (number of sessions, etc.)
  - Stopping/starting instances.
  - Get details (state, active sessions) about apps and instances. Useful for monitoring!
  - Force quit applications/instances
  - And many more (check the wiki)

Wonder's wotaskd/ Monitor

- Wonder's version also keep a backup of SiteConfig.xml after each changes.
- Offers a "bounce" action. This action refuses new sessions for active instances, and starts inactive instances. Useful to start new versions of your apps!

DEMO

Alternatives

- Servlet container
- LEWOAJPAdaptor
- mod_proxy_http

Servlet container

- Pros :
  - More “accepted”.
  - Easier to find an hosting provider (shared hosting and VPS).
  - Can use less RAM.
- Cons :
  - By default, all apps runs in a single VM instance.
  - Some classpath problems.

mod_proxy_http

- Use the standard Apache 2.2 modules, mod_proxy and mod_proxy_httpd
- Pros : no need to run wotaskd and JavaMonitor, so you save RAM. No need to install a servlet container, you only need Apache 2.2 and mod_proxy_http. Useful for VPS or places that won't let you run wotaskd.
- Cons : no watchdog that will restart your instances if they go down. mod_proxy_balancer seems to have problems with Ajax components from Wonder.

Starting an application manually

```
./Uber -WOPort 2008 -WOCachingEnabled YES -WODebuggingEnabled NO -WOOutputPath /Library/
WebObjects/Logs/Uber-1 -WOAutoOpenInBrowser NO -WOAutoOpenClientApplication NO -
WOWorkerThreadCount 8 -WOListenQueueSize 128 -WOWorkerThreadCountMin 16 -
WOWorkerThreadCountMax 256 -WOSessionTimeOut 3600 -WOApplicationName Uber -
WOMonitorEnabled YES -WONoPause YES &
```

mod_proxy_http

```
ProxyRequests On
ProxyMaxForwards 10000
ProxyVia Full
ProxyPass /cgi-bin/WebObjects/Uber.woa http://host:2008/cgi-bin/WebObjects/Uber.woa
ProxyPassReverse /cgi-bin/WebObjects/Uber.woa http://host:2008/cgi-bin/WebObjects/Uber.woa
```

LEWOAJPAdator/ mod_proxy_ajp

- Same concept as mod_proxy_http, but use the more efficient AJP protocol and can work on Apache 1.3 with mod_jk.
- Pros : same as mod_proxy_httpd + can work with Apache 1.3 and is a bit more efficient (try to maintain connections between proxy and application). Works better than mod_proxy_http for load balancing.
- Cons : same as mod_proxy_httpd, and it override WOContext.

Now, to a different subject...

Monitoring!

Monitoring...

- Ok, so you have your server and you can deploy on it...
- But it's your own customers/users who report problems!
- How can you find problems before your customers call you? With monitoring software!

Monitoring...

- Introducting my favorite monitoring tool : Nagios.
- Was called NetSaint a long time ago.
- Been around since 1999.
- Written in C and PERL.
- Works well on OS X since the 1.4 version.
- Now at version 3.

Monitoring...

- Come with a lot of default plugins to check hosts and services like ping, SSH and HTTP.
- Can do remote checks (disk space, # of running processes, etc.) with a plugin and by SNMP.
- Lot of other plugins, for RDMS and other services, available at Nagios Exchange.
- Can write your own plugins, in bash, PERL, C or anything that can run by command line.

Monitoring...

- Easiest way to install it on Mac OS X is with MacPorts (macports.org). MacPorts will install every dependency needed to build and run Nagios.
  - sudo /opt/local/bin/port install nagios
  - sudo /opt/local/bin/port install nrpe
- After installation, you need to add the configuration needed in Apache to use the CGI to access Nagios from a browser.

Monitoring...

- What's the minimum to monitor?
  - Your WO apps and wotaskd.
  - Your RDBMS (at least check if it's running!).
  - Ping, disk space, load and SSH.

DEMO

Monitoring...

- Your new best friend : check_http
- This plugin will check if a HTTP connection can be established, and optionally check if specific content is sent back.
- Checking for specific content is useful to make sure that it's not a WO adaptor message (« The requested application was not found », etc.)
- Even more useful is to use it with Wonder's JavaMonitor direct actions!

DEMO

Monitoring...

- RDBMS plugins are your friends too!
  - For Oracle, can check tablespace usage and do a login.
  - For MySQL, can check if the slave thread is running, and do a login.
  - For PostgreSQL, can check response time and check if a database is present.
  - Plugins for other RDBMS are also available on Nagios Exchange.

Monitoring...

- Other nice features :
    - On call rotation. Have different staff on call on different time periods? No problem!
    - Can create time periods for checks and alerts. No need to get alerts at 3am for less important stuff!
    - Dependencies. A switch is down? Nagios won't complain about gear connected to this switch.

But wait... What about JMX?

JMX

JMX (Java Management Extensions) let you monitor and collect statistics about various parts of the JVM instance. For example, you can monitor heap space usage, maximum heap space, number of threads.

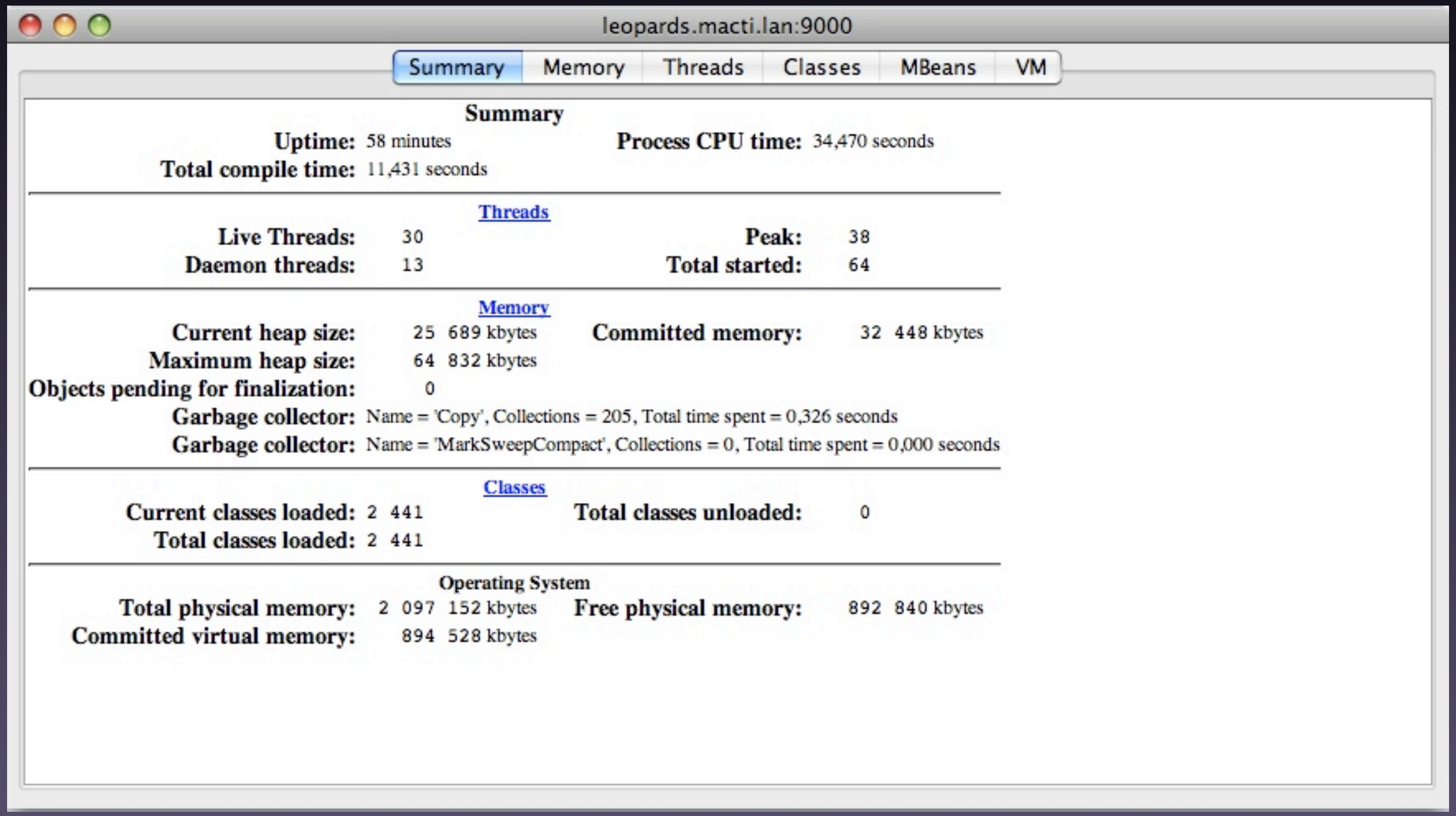
JMX

You enable JMX by adding a couple of arguments to the JVM :

-Dcom.sun.management.jmxremote.port=9000
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false

And you connect with *jconsole*

Sample jconsole session



Would be nice to monitor JMX data with Nagios...

Introducing check_jmx

- A new best friend : check_jmx
- This Nagios plugin is written in Java, it's even shipping as a Eclipse project!
- You can check any value from JMX, as long it's a number.
- It also include a plugin for Munin, another popular monitoring system.

JMX

Sample : check if memory is higher than 52 MB (warning level) or higher than 58 MB (critical level).

./check_jmx -U service:jmx:rmi:///jndi/rmi://leopards.macti.lan:9000/jmxrmi -O java.lang:type=Memory -A HeapMemoryUsage -K used -I HeapMemoryUsage -J used -w 54525952 -c 60817408

Result :

JMX OK 26403168

I think WO supports JMX...

Yes, you can get WO specific stats with WO 5.4!

JMX support in WO 5.4

- The output of WOStats is also available from JMX.
- ... But it's only for WO 5.4 and you must add code to Application.java
- You can add your own MBean to monitor too! Might be useful to expose values to Nagios by JMX.

Sample statistics

Attributes | Operations | Notifications | Info

Name	Value
ApplicationLastAccessTime	
ApplicationlastAccessTime	1250110879199
AverageCATransactionTime	30886
AverageDATransactionTime	0
AverageIdleTime	66300
AverageRequestsPerSession	2.0
AverageSessionMemory	{Free Memory=0, Total Memory=0}
AverageTransactionTime	30886
AverageWSTransactionTime	0
DirectActionStatistics	{}
LastSessionStatistics	[org.wocommunity.website.components.Main]
MemoryUsage	{Free Memory=6450904, Total Memory=22835200}
MovingAverageIdleTime	66300
MovingAverageRequestsPerSession	2.0
MovingAverageSessionLife	177587
MovingAverageTransactionTime	30886
PagesStatistics	{org.wocommunity.website.components.Main={Served=...
PathsStatistics	{org.wocommunity.website.components.Main=1}
SessionMovingAverageSampleSize	
StatisticStoreSessionMax	50
TransactionMovingAverageSampleSize	100
WebServiceStatistics	{}

Application.java

```
    import com.webobjects.appserver.WOStatisticsStore;

public Application() throws IllegalAccessException {
    NSLog.out.appendln("Welcome to " + this.name() + " !");

    registerMBean((Object)statisticsStore(), getJMXDomain(),
                  WOStatisticsStore.class.getName());
}
```

Resources

- Nagios : www.nagios.org
- JMX and WO : http://developer.apple.com/documentation/webobjects/Deployment/Deploying_Applications/JMXMonitoring/JMXMonitoring.html
- check_jmx : http://www.monitoringexchange.org/cgi-bin/page.cgi?g=Detailed%2F3002.html;d=1
- http://developer.apple.com/documentation/WebObjects/Deployment/Deploying_Applications/Deploying_Applications.pdf
- LEWOAJPAdaptor/LEWOStuff : http://homepage.mac.com/andrewlindesay/le/
- JSCH : http://www.jcraft.com/jsch/
- Webmin : www.webmin.com