Auditoria e Qualidade de Sistemas

Prof. Edgard Davidson C. Cardoso

Qualidade de Sistemas

PROFILER

una

Problemas Comuns

- No seu projeto
  - Identificar como está o consumo de memória
  - Identificar como está o consumo do processador
  - Identificar como está o comportamentos das threads

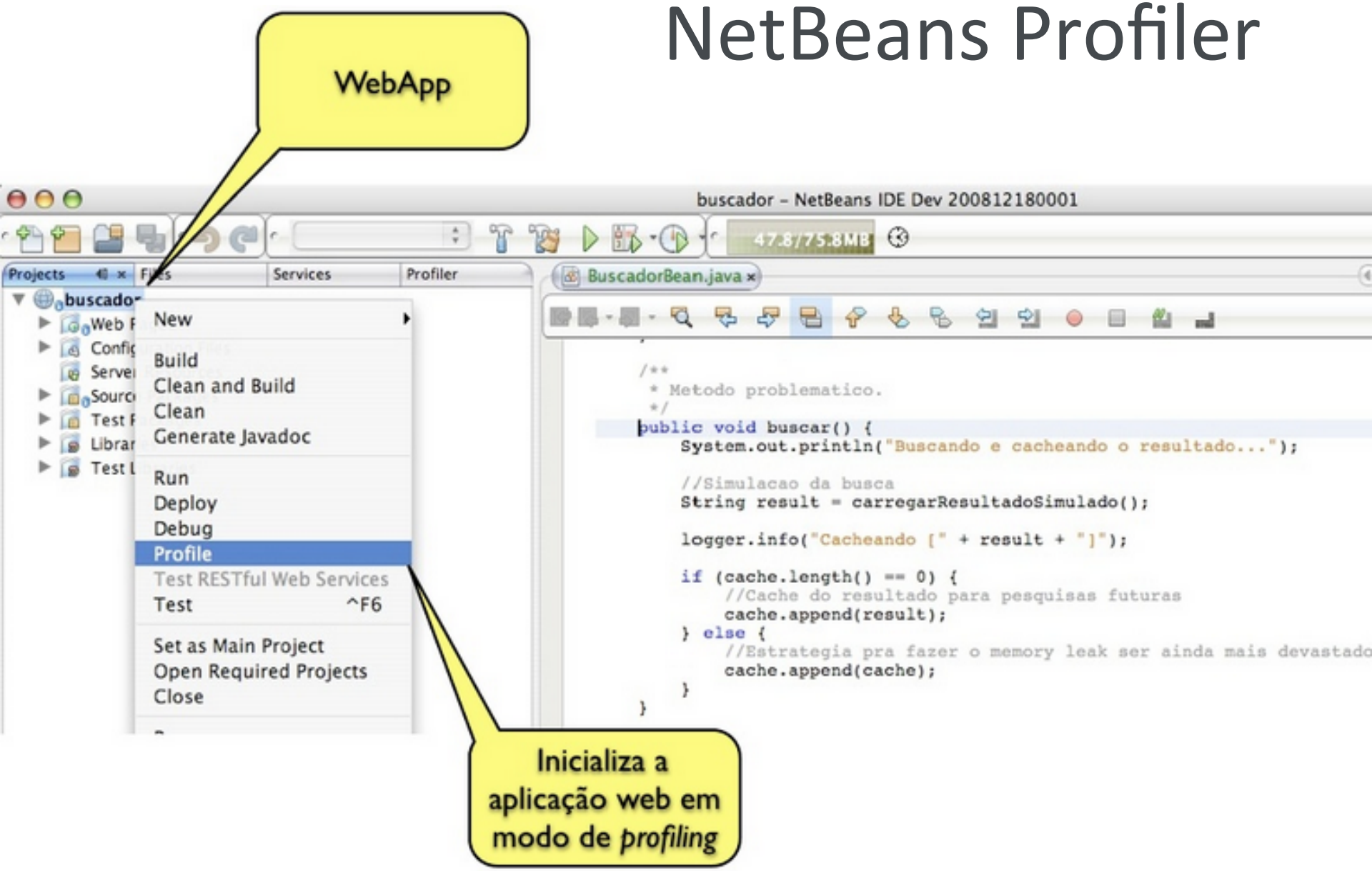
Ferramentas de Profiling

- NetBeans Profiler
- Jboss Profiler
- Jprofiler
- Yourkit

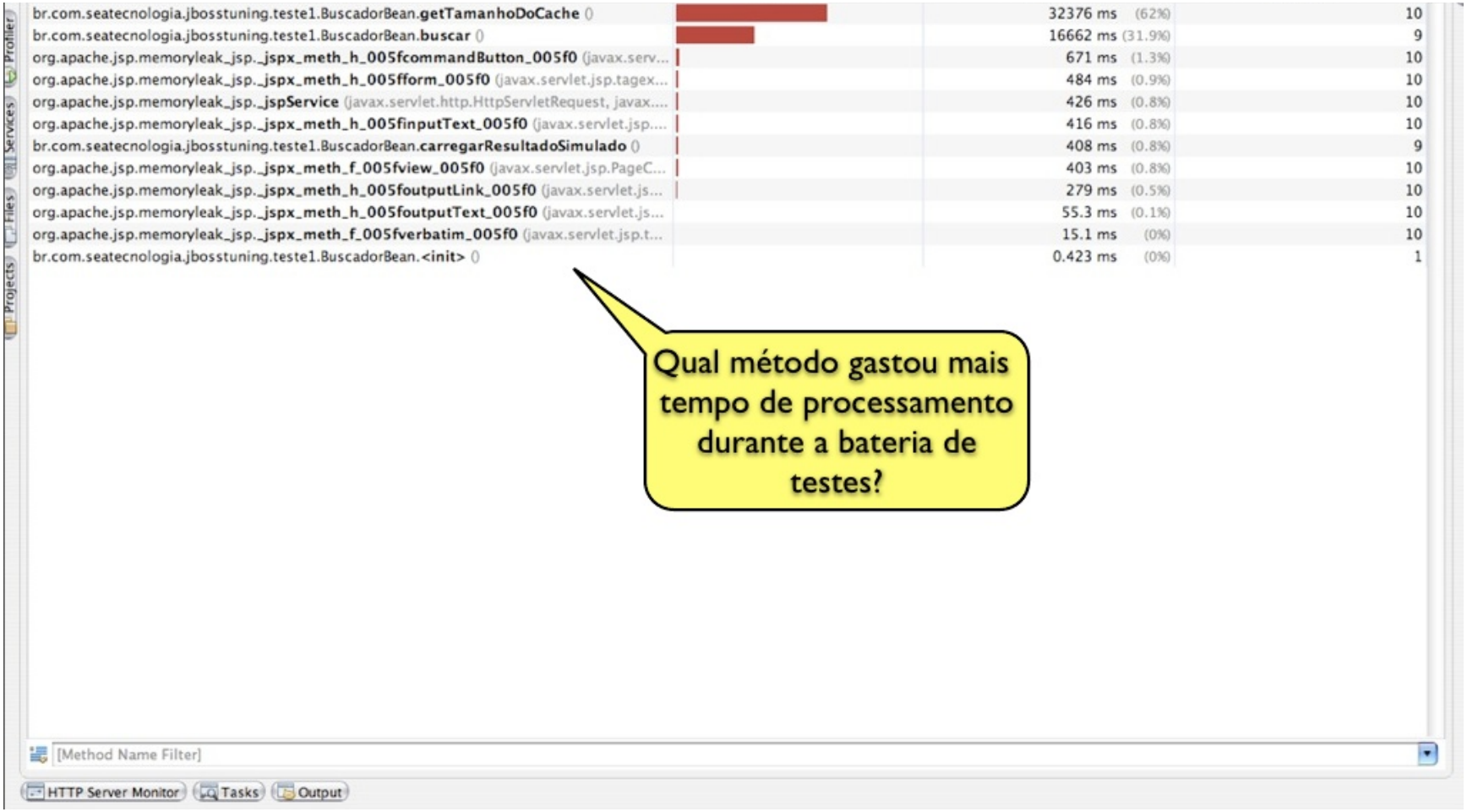
NetBeans Profiler

- Demonstra o comportamento de sua aplicação em tempo de execução
  - Tamanho do heap de memória,
  - Estatísticas do GC
  - Dados de threas
  - Tempo em que cada método utiliza a CPU

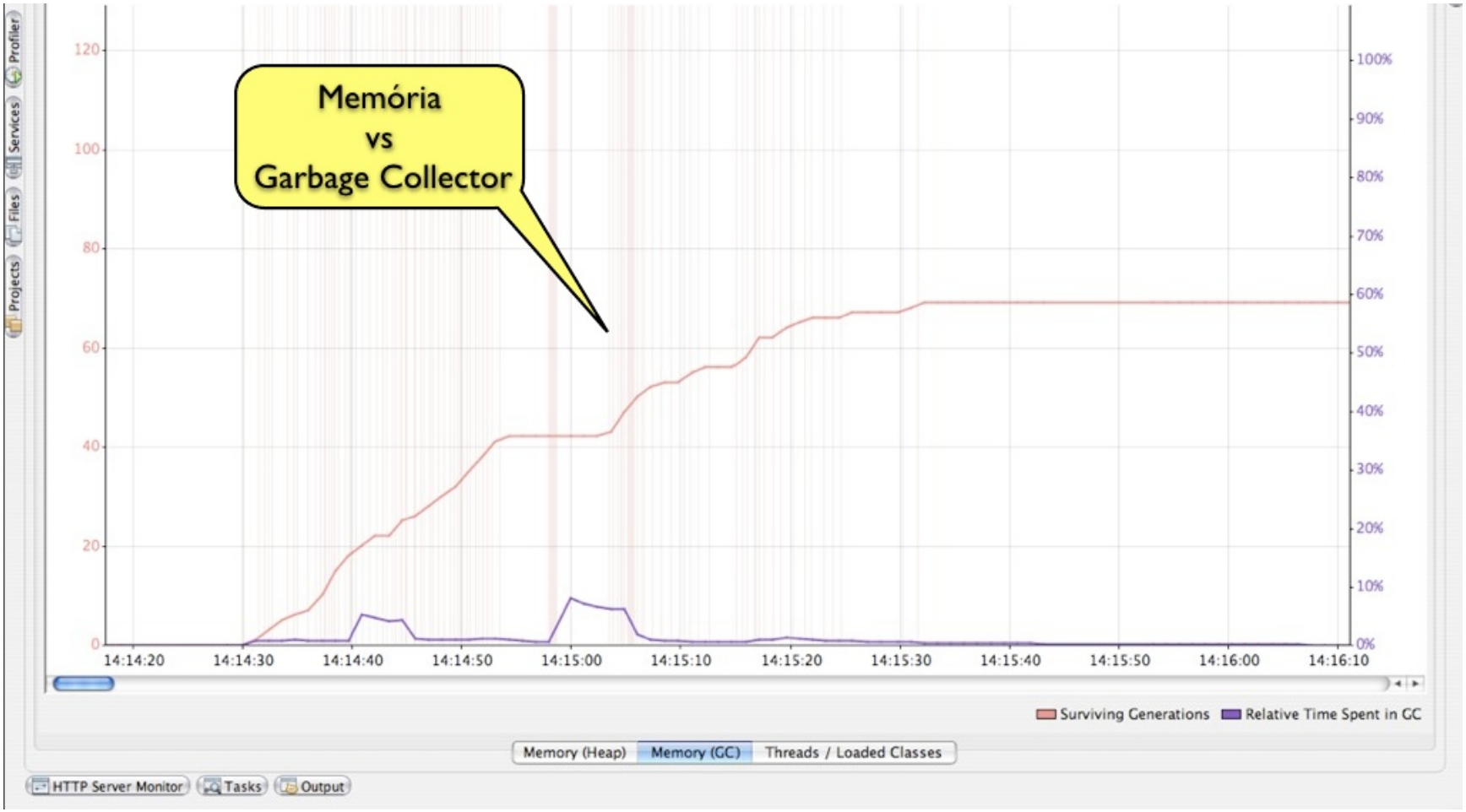
NetBeans Profiler



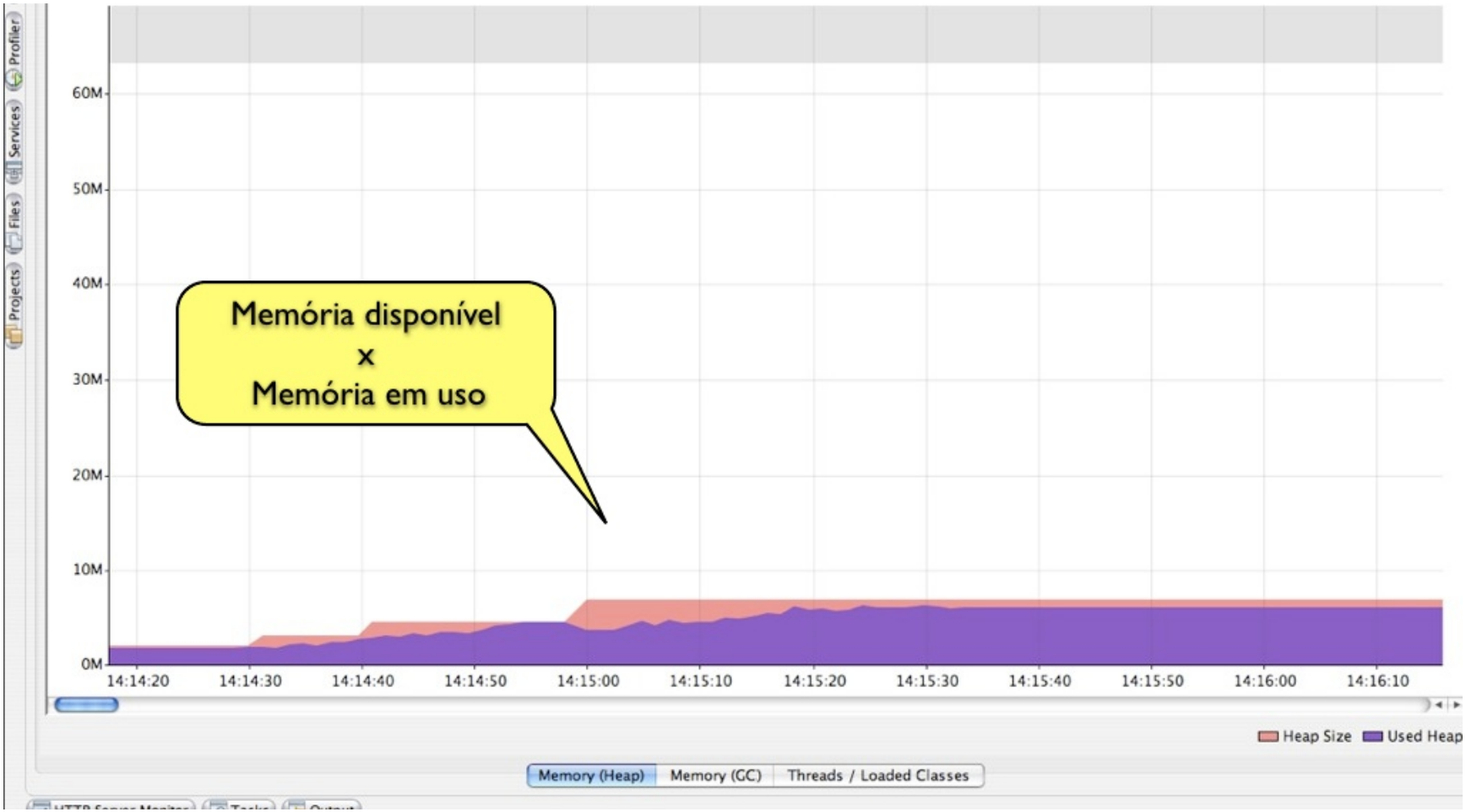
NetBeans Profiler



NetBeans Profiler

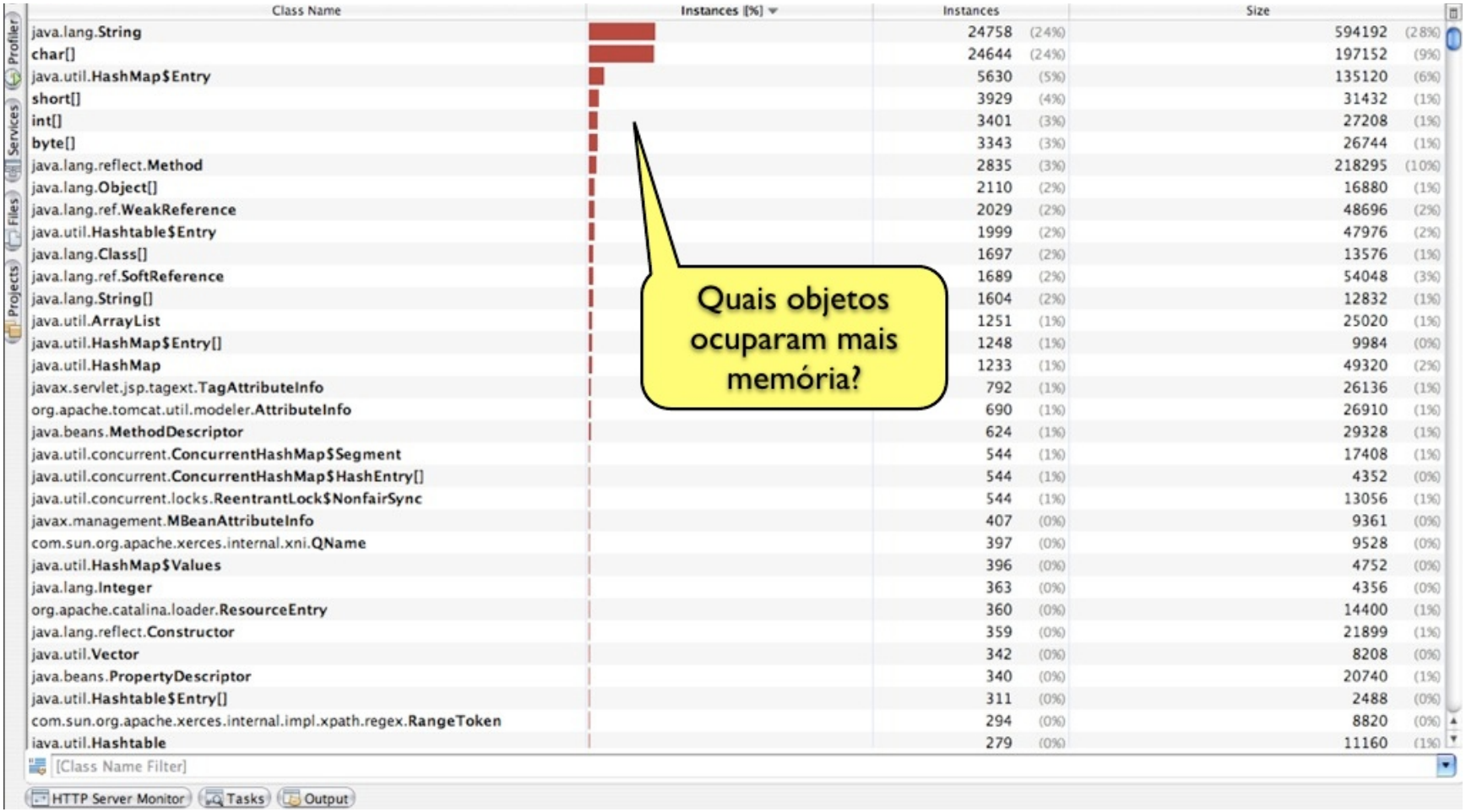


NetBeans Profiler



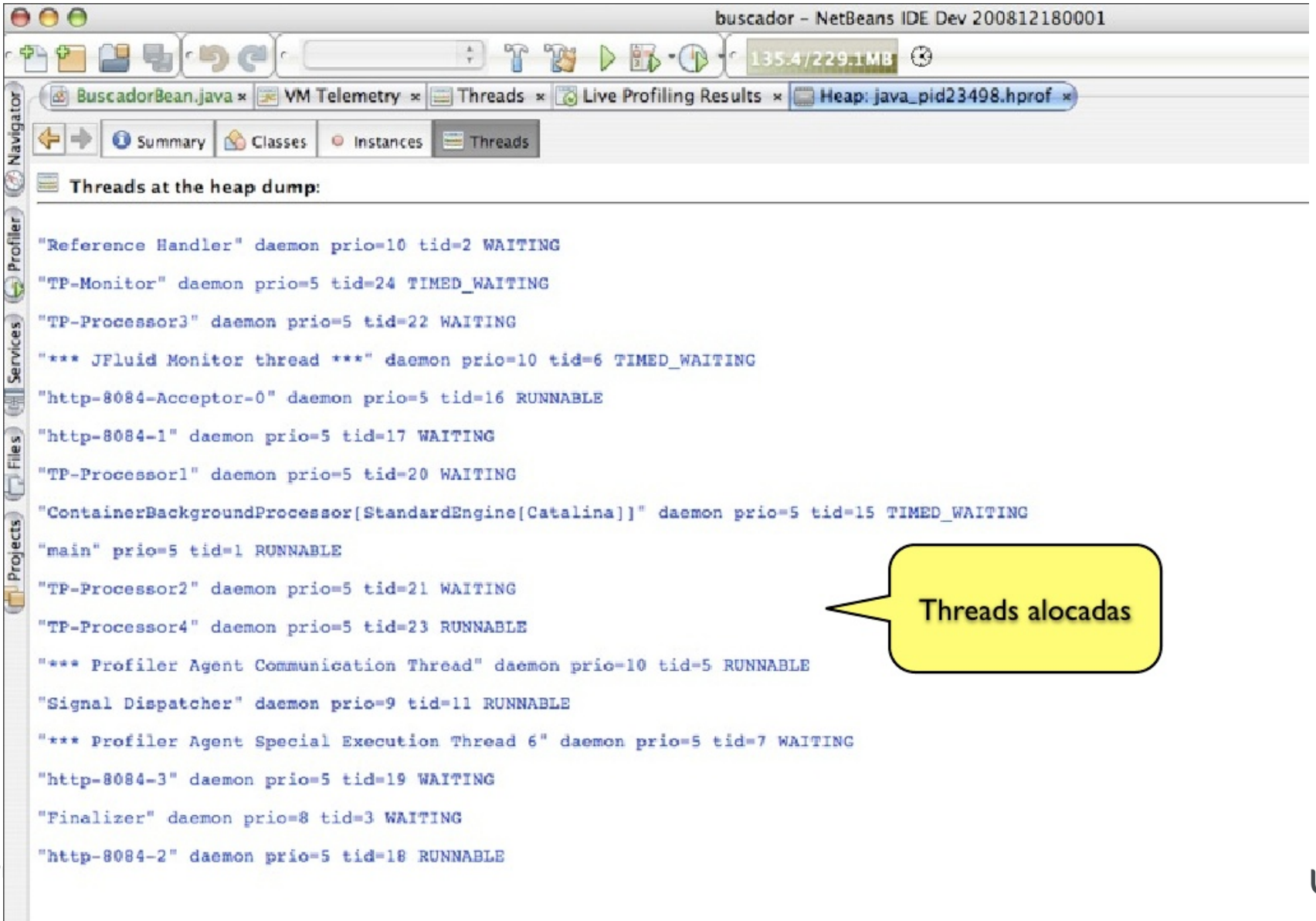
NetBeans Profiler

NetBeans Profiler



Class Name	Instances [%]	Instances		Size	
java.lang.**String**		24758	(24%)	594192	(28%)
char[]		24644	(24%)	197152	(9%)
java.util.**HashMap$Entry**		5630	(5%)	135120	(6%)
short[]		3929	(4%)	31432	(1%)
int[]		3401	(3%)	27208	(1%)
byte[]		3343	(3%)	26744	(1%)
java.lang.reflect.**Method**		2835	(3%)	218295	(10%)
java.lang.**Object[]**		2110	(2%)	16880	(1%)
java.lang.ref.**WeakReference**		2029	(2%)	48696	(2%)
java.util.**Hashtable$Entry**		1999	(2%)	47976	(2%)
java.lang.**Class[]**		1697	(2%)	13576	(1%)
java.lang.ref.**SoftReference**		1689	(2%)	54048	(3%)
java.lang.**String[]**		1604	(2%)	12832	(1%)
java.util.**ArrayList**		1251	(1%)	25020	(1%)
java.util.**HashMap$Entry[]**		1248	(1%)	9984	(0%)
java.util.**HashMap**		1233	(1%)	49320	(2%)
javax.servlet.jsp.tagext.**TagAttributeInfo**		792	(1%)	26136	(1%)
org.apache.tomcat.util.modeler.**AttributeInfo**		690	(1%)	26910	(1%)
java.beans.**MethodDescriptor**		624	(1%)	29328	(1%)
java.util.concurrent.**ConcurrentHashMap$Segment**		544	(1%)	17408	(1%)
java.util.concurrent.**ConcurrentHashMap$HashEntry[]**		544	(1%)	4352	(0%)
java.util.concurrent.locks.**ReentrantLock$NonfairSync**		544	(1%)	13056	(1%)
javax.management.**MBeanAttributeInfo**		407	(0%)	9361	(0%)
com.sun.org.apache.xerces.internal.xni.**QName**		397	(0%)	9528	(0%)
java.util.**HashMap$Values**		396	(0%)	4752	(0%)
java.lang.**Integer**		363	(0%)	4356	(0%)
org.apache.catalina.loader.**ResourceEntry**		360	(0%)	14400	(1%)
java.lang.reflect.**Constructor**		359	(0%)	21899	(1%)
java.util.**Vector**		342	(0%)	8208	(0%)
java.beans.**PropertyDescriptor**		340	(0%)	20740	(1%)
java.util.**Hashtable$Entry[]**		311	(0%)	2488	(0%)
com.sun.org.apache.xerces.internal.impl.xpath.regex.**RangeToken**		294	(0%)	8820	(0%)
java.util.**Hashtable**		279	(0%)	11160	(1%)

NetBeans Profiler



una

NetBeans Profiler

```
        javax.faces.webapp.FacesServlet.service(FacesServlet.java:256)
        org.netbeans.modules.web.monitor.server.MonitorFilter.doFilter(MonitorFilter.java:390)

root cause

org.apache.jasper.el.JspELException: /memoryleak.jsp(37,24) '#{buscador.tamanhoDoCache}' Error reading 'tamanhoDoCache' or
        org.apache.jasper.el.JspValueExpression.getValue(JspValueExpression.java:107)
        javax.faces.component.UIOutput.getValue(UIOutput.java:173)
        com.sun.faces.renderkit.html_basic.HtmlBasicInputRenderer.getValue(HtmlBasicInputRenderer.java:189)
        com.sun.faces.renderkit.html_basic.HtmlBasicRenderer.getCurrentValue(HtmlBasicRenderer.java:320)
        com.sun.faces.renderkit.html_basic.HtmlBasicRenderer.encodeEnd(HtmlBasicRenderer.java:200)
        javax.faces.component.UIComponentBase.encodeEnd(UIComponentBase.java:836)
        javax.faces.component.UIComponent.encodeAll(UIComponent.java:896)
        javax.faces.render.Renderer.encodeChildren(Renderer.java:137)
        javax.faces.component.UIComponentBase.encodeChildren(UIComponentBase.java:812)
        javax.faces.component.UIComponent.encodeAll(UIComponent.java:886)
        javax.faces.component.UIComponent.encodeAll(UIComponent.java:892)
        com.sun.faces.application.ViewHandlerImpl.doRenderView(ViewHandlerImpl.java:245)
        com.sun.faces.application.ViewHandlerImpl.renderView(ViewHandlerImpl.java:176)
        com.sun.faces.lifecycle.RenderResponsePhase.execute(RenderResponsePhase.java:10
        com.sun.faces.lifecycle.LifecycleImpl.phase(LifecycleImpl.java:251)
        com.sun.faces.lifecycle.LifecycleImpl.render(LifecycleImpl.java:144)
        javax.faces.webapp.FacesServlet.service(FacesServlet.java:245)
        org.netbeans.modules.web.monitor.server.MonitorFilter.doFi    r(MonitorFilter.java:390)

root cause

java.lang.OutOfMemoryError: Java heap space
        java.lang.String.toCharArray(String.java:2514)
        br.com.seatecnologia.jbosstuning.testel.BuscadorBean.getTamanhoDoCache(BuscadorBean.java:39)
        sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
        sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
        java.lang.reflect.Method.invoke(Method.java:585)
        javax.el.BeanELResolver.getValue(BeanELResolver.java:62)
        javax.el.CompositeELResolver.getValue(CompositeELResolver.java:53)
        com.sun.faces.el.FacesCompositeELResolver.getValue(FacesCompositeELResolver.java:64)
```

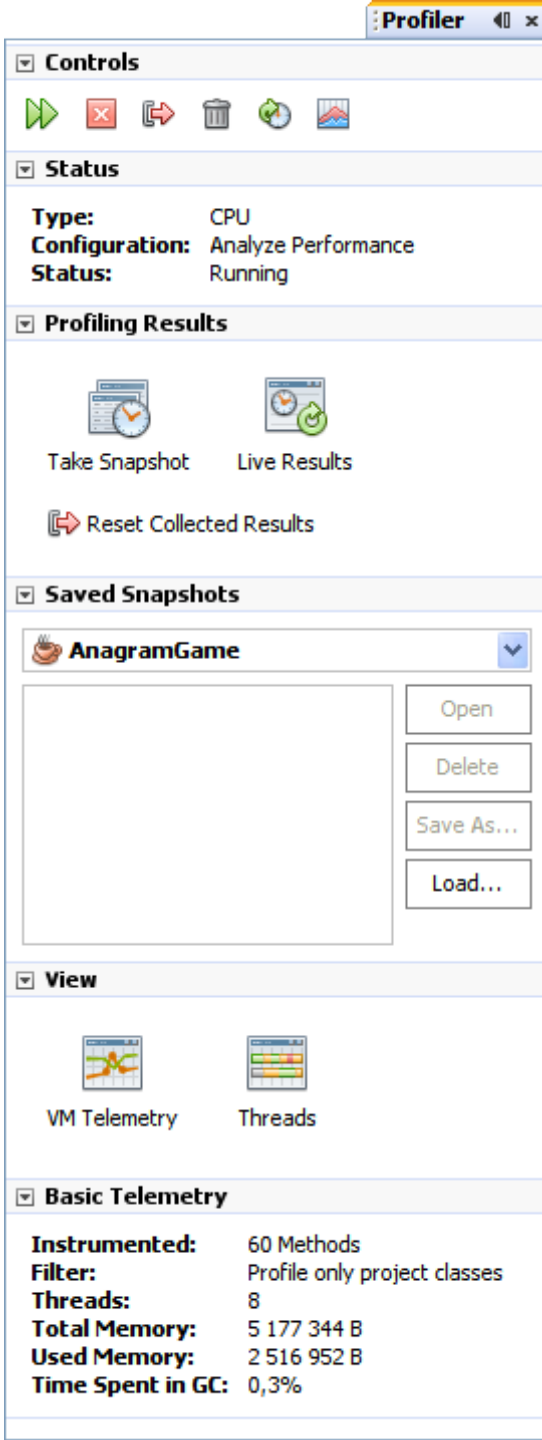
Acabou a memória disponível.

NetBeans Profiler

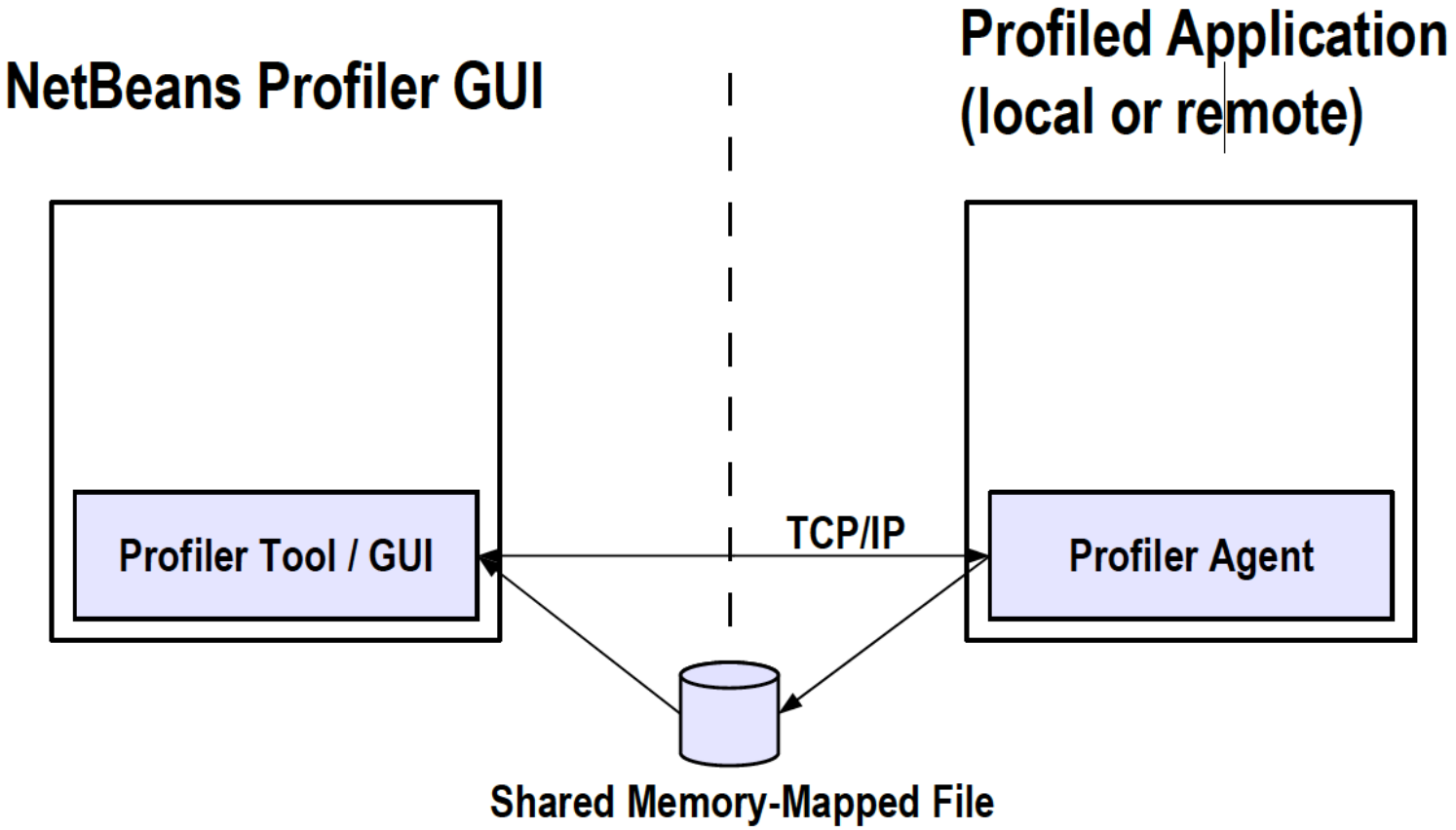
una

NetBeans Profiler

- Funcionalidades compleas de Java profiler
  - Monitoramento da aplicação (Threads, VM Telemetria)
  - Profiling de performance (CPU)
  - Profiling de memória,
  - Detecção de Memory Leaks

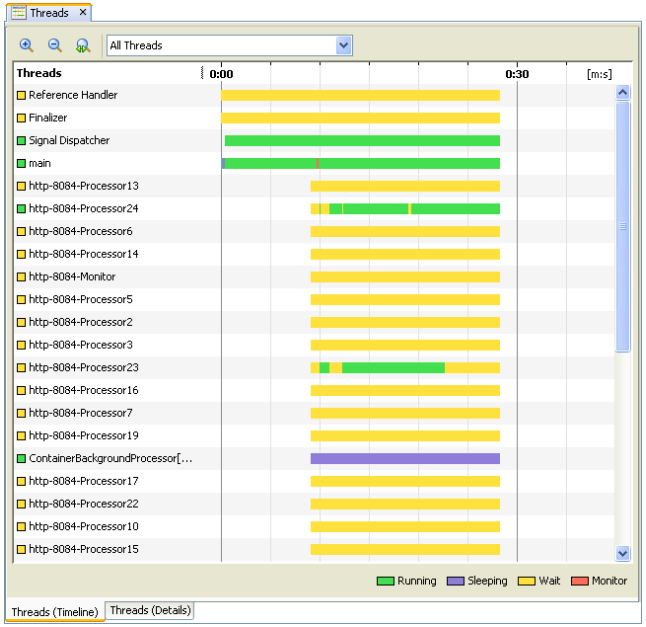


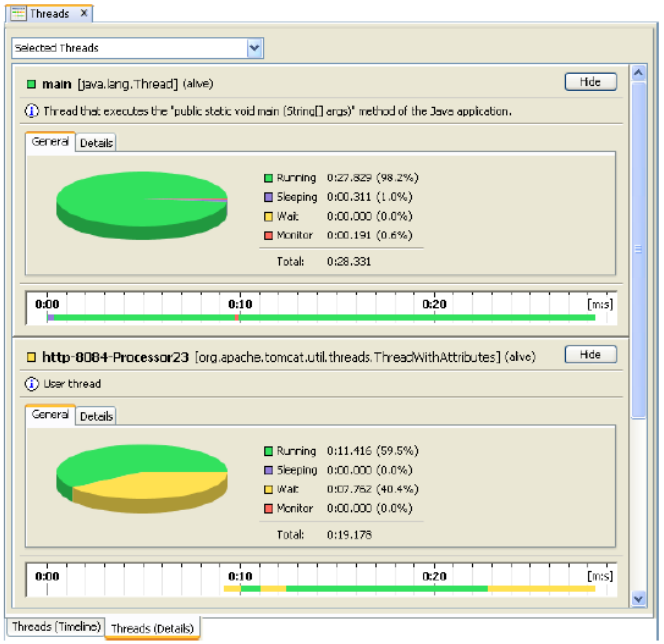
Arquitetura



Monitoramento de Threads

- Linha do tempo de Threads
  - Nome da Thread, classe
  - Status da Thread no tempo
- Detalhes da Thread
  - Estatística de Atividade
  - Timestamp para cada estado
  - Descrição de threads de systema

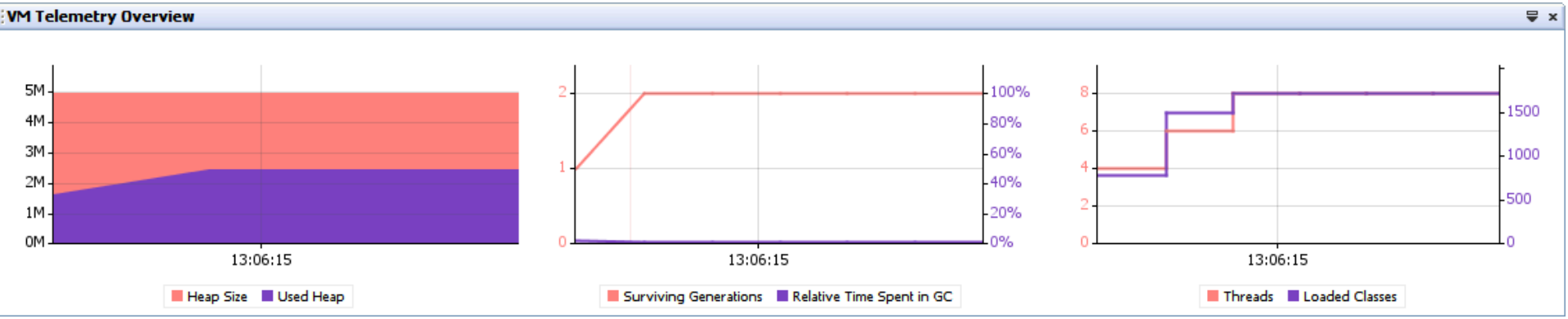




una

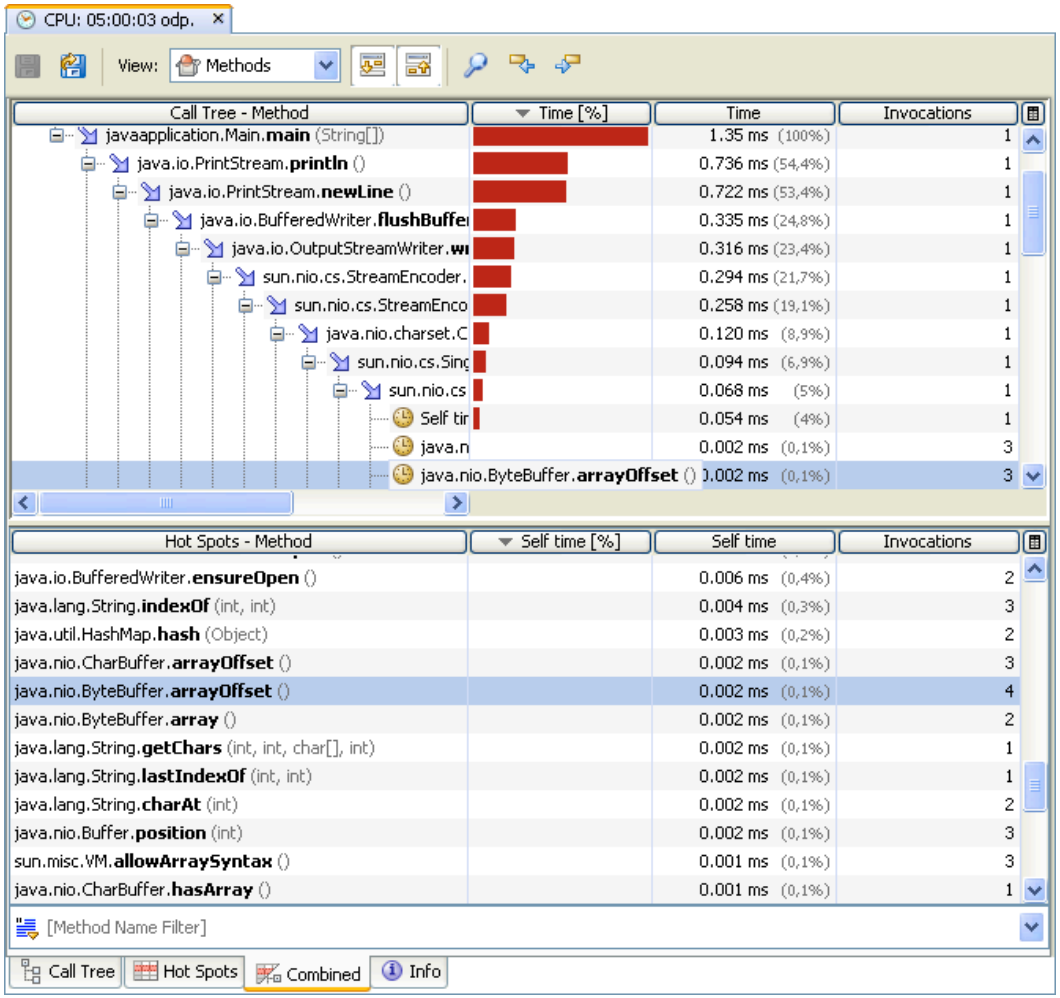
Monitoramento de Telemetria da VM

- Visualização da telemetria da máquina virtual
  - Heap alocada, heap usada
  - Número de Threads executando na VM
  - Tempo relativo gasto pelo garbage collection
  - Geração de métricas de detecção de memory leak



CPU Profiling

- Profiling da aplicação ou parte da aplicação
- Tempo gasto por cada método
- CPU snapshots
- Back Traces



Profiling de Memória

- rastreamento de alocações e vida de objetos
- rastreamentos de todos objetos
- Gravação de stack traces