



Auditoria e Qualidade de Sistemas

Prof. Edgard Davidson C. Cardoso



Qualidade de Sistemas

PROFILER



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 threads
 - Tempo em que cada método utiliza a CPU

NetBeans Profiler

WebApp

Inicializa a aplicação web em modo de profiling

```
/**  
 * Metodo problematico.  
 */  
public void buscar() {  
    System.out.println("Buscando e cacheando o resultado...");  
  
    //Simulacao da busca  
    String result = carregarResultadosSimulado();  
  
    logger.info("Cacheando [" + result + "]");  
  
    if (cache.length() == 0) {  
        //Cache do resultado para pesquisas futuras  
        cache.append(result);  
    } else {  
        //Estrategia pra fazer o memory leak ser ainda mais devastador  
        cache.append(cache);  
    }  
}
```

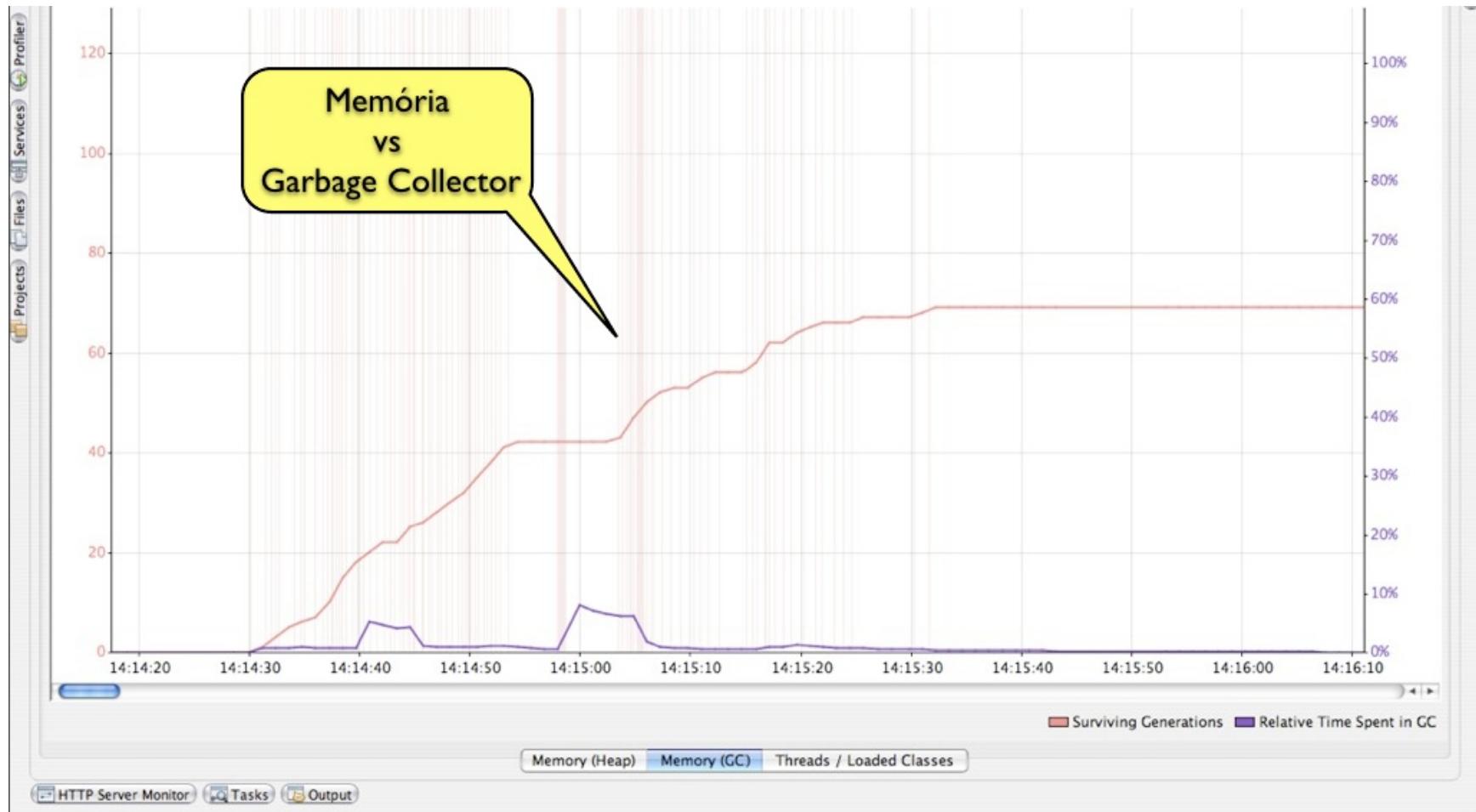


NetBeans Profiler



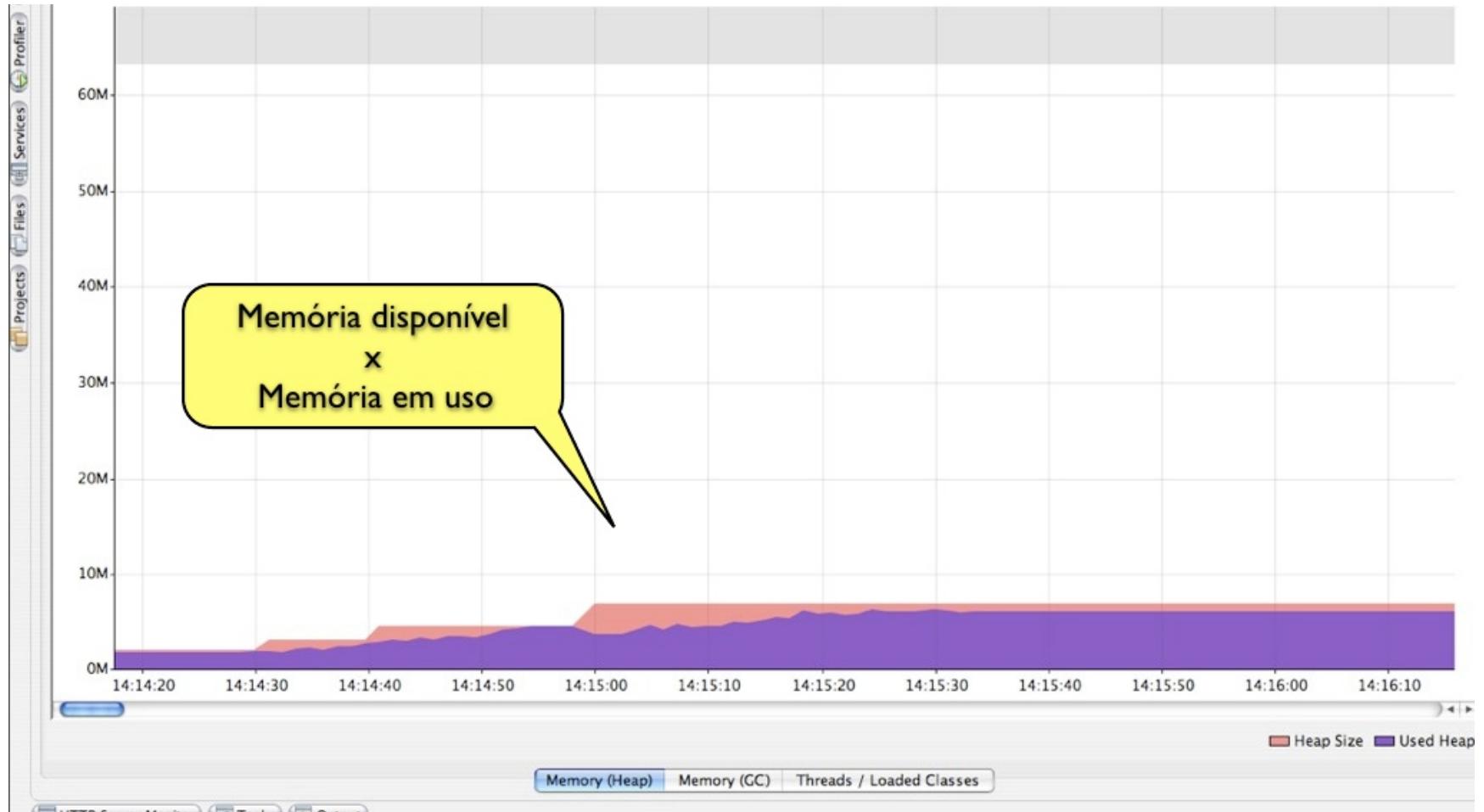


NetBeans Profiler





NetBeans Profiler



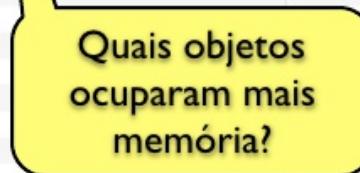


NetBeans Profiler





NetBeans Profiler



Class Name	Instances [%]	Instances	Size
java.lang.String	2.4%	24758	594192 (2.8%)
char[]	2.4%	24644	197152 (9%)
java.util.HashMap\$Entry	5%	5630	135120 (6%)
short[]	4%	3929	31432 (1%)
int[]	3%	3401	27208 (1%)
byte[]	3%	3343	26744 (1%)
java.lang.reflect.Method	3%	2835	218295 (10%)
java.lang.Object[]	2%	2110	16880 (1%)
java.lang.ref.WeakReference	2%	2029	48696 (2%)
java.util.Hashtable\$Entry	2%	1999	47976 (2%)
java.lang.Class[]	2%	1697	13576 (1%)
java.lang.ref.SoftReference	2%	1689	54048 (3%)
java.lang.String[]	2%	1604	12832 (1%)
java.util.ArrayList	1%	1251	25020 (1%)
java.util.HashMap\$Entry[]	1%	1248	9984 (0%)
java.util.HashMap	1%	1233	49320 (2%)
javax.servlet.jsp.tagext.TagAttributeInfo	1%	792	26136 (1%)
org.apache.tomcat.util.modeler.AttributeInfo	1%	690	26910 (1%)
java.beans.MethodDescriptor	1%	624	29328 (1%)
java.util.concurrent.ConcurrentHashMap\$Segment	1%	544	17408 (1%)
java.util.concurrent.ConcurrentHashMap\$HashEntry[]	1%	544	4352 (0%)
java.util.concurrent.locks.ReentrantLock\$NonfairSync	1%	544	13056 (1%)
javax.management.MBeanAttributeInfo	0%	407	9361 (0%)
com.sun.org.apache.xerces.internal.xni.QName	0%	397	9528 (0%)
java.util.HashMap\$Values	0%	396	4752 (0%)
java.lang.Integer	0%	363	4356 (0%)
org.apache.catalina.loader.ResourceEntry	0%	360	14400 (1%)
java.lang.reflect.Constructor	0%	359	21899 (1%)
java.util.Vector	0%	342	8208 (0%)
java.beans.PropertyDescriptor	0%	340	20740 (1%)
java.util.Hashtable\$Entry[]	0%	311	2488 (0%)
com.sun.org.apache.xerces.internal.impl.xpath.regex.RangeToken	0%	294	8820 (0%)
java.utilHashtable	0%	279	11160 (1%)



NetBeans Profiler

The screenshot shows the NetBeans IDE interface with the title bar "buscador - NetBeans IDE Dev 200812180001". The toolbar includes icons for file operations, navigation, and profiling. The menu bar has "File", "Edit", "Tools", "View", "Project", "Services", "Profiler", and "Help". The left sidebar has tabs for "Projects", "Files", "Services", "Profiler" (which is selected), and "Navigator". The main window displays a "Threads at the heap dump:" section with a list of threads:

```
"Reference Handler" daemon prio=10 tid=2 WAITING  
"TP-Monitor" daemon prio=5 tid=24 TIMED_WAITING  
"TP-Processor3" daemon prio=5 tid=22 WAITING  
"*** JFluid Monitor thread ***" daemon prio=10 tid=6 TIMED_WAITING  
"http-8084-Acceptor-0" daemon prio=5 tid=16 RUNNABLE  
"http-8084-1" daemon prio=5 tid=17 WAITING  
"TP-Processor1" daemon prio=5 tid=20 WAITING  
"ContainerBackgroundProcessor[StandardEngine[Catalina]]" daemon prio=5 tid=15 TIMED_WAITING  
"main" prio=5 tid=1 RUNNABLE  
"TP-Processor2" daemon prio=5 tid=21 WAITING  
"TP-Processor4" daemon prio=5 tid=23 RUNNABLE  
"*** Profiler Agent Communication Thread" daemon prio=10 tid=5 RUNNABLE  
"Signal Dispatcher" daemon prio=9 tid=11 RUNNABLE  
"*** Profiler Agent Special Execution Thread 6" daemon prio=5 tid=7 WAITING  
"http-8084-3" daemon prio=5 tid=19 WAITING  
"Finalizer" daemon prio=8 tid=3 WAITING  
"http-8084-2" daemon prio=5 tid=18 RUNNABLE
```

A yellow callout bubble points to the word "Threads" in the list, with the text "Threads alocadas" (Allocated Threads) inside.



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.doFilter(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

```
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:206)
at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:233)
at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:191)
at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:128)
at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:102)
at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:109)
at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:286)
at org.apache.coyote.http11.Http11Processor.process(Http11Processor.java:845)
at org.apache.coyote.http11.Http11Protocol$Http11ConnectionHandler.process(Http11Protocol.java:583)
at org.apache.tomcat.util.net.JIoEndpoint$Worker.run(JIoEndpoint.java:447)
at java.lang.Thread.run(Thread.java:613)
Caused by: org.apache.jasper.el.JspELException: /memoryleak.jsp(37,24) '#{buscador.tamanhoDoCache}' Error reading 'tamanhoDoCache' on type br.
at org.apache.jasper.el.JspValueExpression.getValue(JspValueExpression.java:107)
at javax.faces.component.UIOutput.getValue(UIOutput.java:173)
... 30 more
Caused by: java.lang.OutOfMemoryError: Java heap space
at java.lang.String.toCharArray(String.java:2514)
at br.com.seatecnologia.jbosstuning.teste1.BuscadorBean.getTamanhoDoCache(BuscadorBean.java:39)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
at java.lang.reflect.Method.invoke(Method.java:585)
at javax.el.BeanELResolver.getValue(BeanELResolver.java:62)
at javax.el.CompositeELResolver.getValue(CompositeELResolver.java:53)
at com.sun.faces.el.FacesCompositeELResolver.getValue(FacesCompositeELResolver.java:64)
at org.apache.el.parser.AstValue.getValue(AstValue.java:118)
at org.apache.el.ValueExpressionImpl.getValue(ValueExpressionImpl.java:186)
at org.apache.jasper.el.JspValueExpression.getValue(JspValueExpression.java:101)
at javax.faces.component.UIOutput.getValue(UIOutput.java:173)
at com.sun.faces.renderkit.html_basic.HtmlBasicInputRenderer.getValue(HtmlBasicInputRenderer.java:189)
at com.sun.faces.renderkit.html_basic.HtmlBasicRenderer.getCurrentValue(HtmlBasicRenderer.java:320)
at com.sun.faces.renderkit.html_basic.HtmlBasicRenderer.encodeEnd(HtmlBasicRenderer.java:200)
at javax.faces.component.UICOMPONENTBase.encodeEnd(UICOMPONENTBase.java:836)
at javax.faces.component.UIComponent.encodeAll(UIComponent.java:896)
at javax.faces.render.Renderer.encodeChildren(Renderer.java:137)
at javax.faces.component.UICOMPONENTBase.encodeChildren(UICOMPONENTBase.java:812)
at javax.faces.component.UIComponent.encodeAll(UIComponent.java:886)
at javax.faces.component.UIComponent.encodeAll(UIComponent.java:892)
at com.sun.faces.application.ViewHandlerImpl.doRenderView(ViewHandlerImpl.java:245)
at com.sun.faces.application.ViewHandlerImpl.renderView(ViewHandlerImpl.java:176)
at com.sun.faces.lifecycle.RenderResponsePhase.execute(RenderResponsePhase.java:106)
at com.sun.faces.lifecycle.LifecycleImpl.phase(LifecycleImpl.java:251)
at com.sun.faces.lifecycle.LifecycleImpl.render(LifecycleImpl.java:144)
at javax.faces.webapp.FacesServlet.service(FacesServlet.java:245)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:290)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:206)
at org.netbeans.modules.web.monitor.server.MonitorFilter.doFilter(MonitorFilter.java:390)
```

Mesmo erro no log
do Tomcat do
NetBeans



NetBeans Profiler

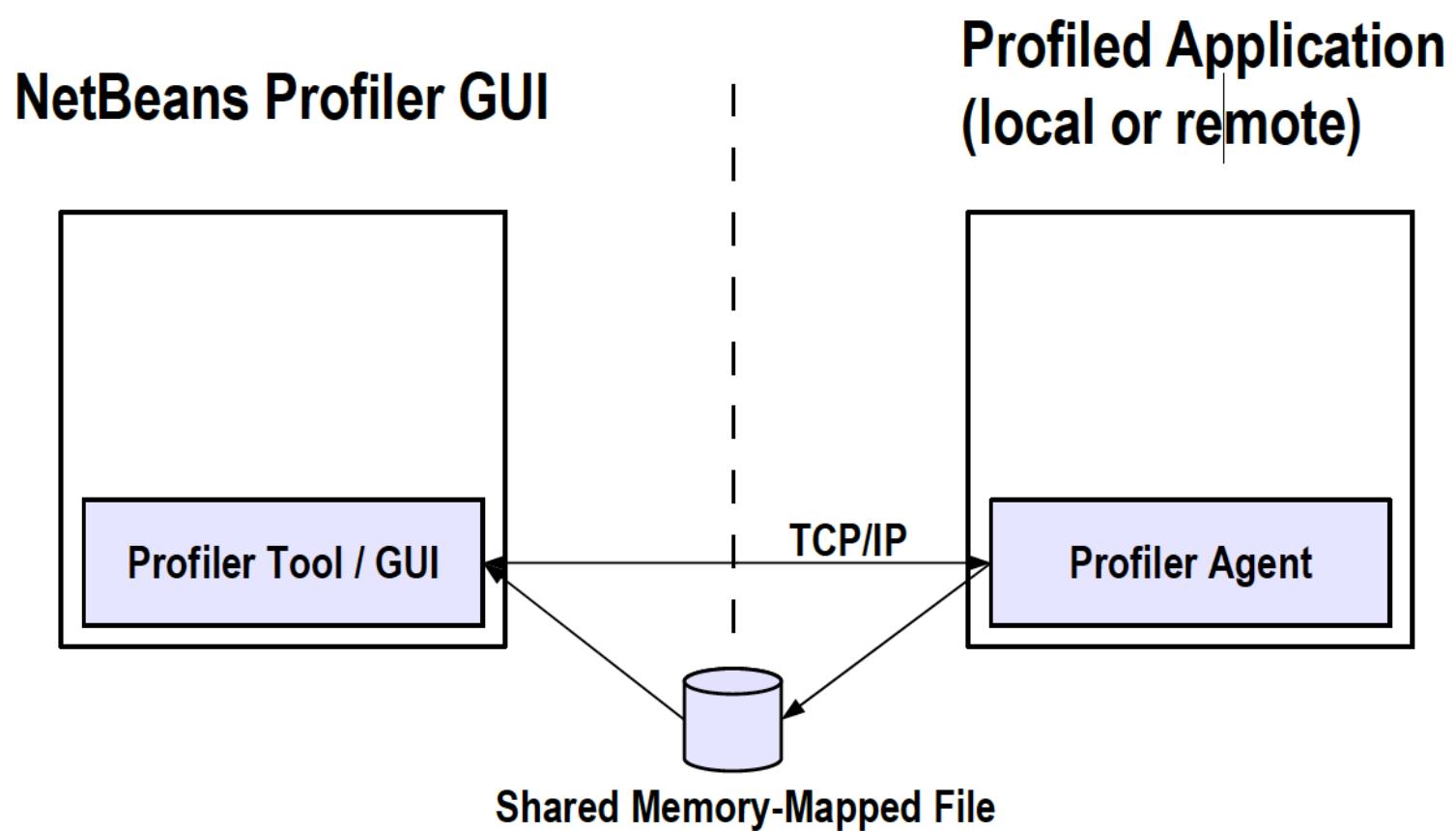
- Funcionalidades compleas de Java profiler
 - Monitoramento da aplicação (Threads, VM Telemetria)
 - Profiling de performance (CPU)
 - Profiling de memória,
 - Detecção de Memory Leaks

The screenshot shows the NetBeans Profiler interface with several panels:

- Controls**: Includes icons for New Project, Close, Open, Delete, Refresh, and Help.
- Status**: Shows Type: CPU, Configuration: Analyze Performance, and Status: Running.
- Profiling Results**: Contains icons for Take Snapshot and Live Results, and a Reset Collected Results button.
- Saved Snapshots**: A list showing "AnagramGame" with options to Open, Delete, Save As..., and Load... .
- View**: Icons for VM Telemetry and Threads.
- Basic Telemetry**: Statistics including Instrumented: 60 Methods, Filter: Profile only project classes, Threads: 8, Total Memory: 5 177 344 B, Used Memory: 2 516 952 B, and Time Spent in GC: 0,3%.



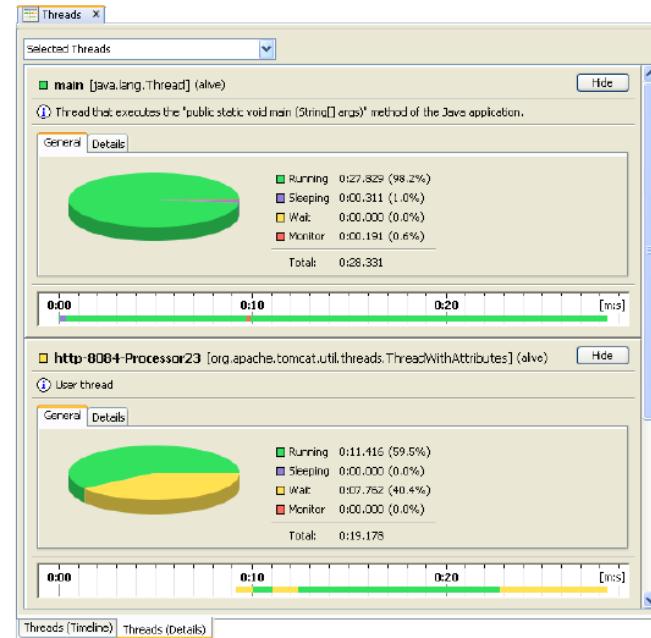
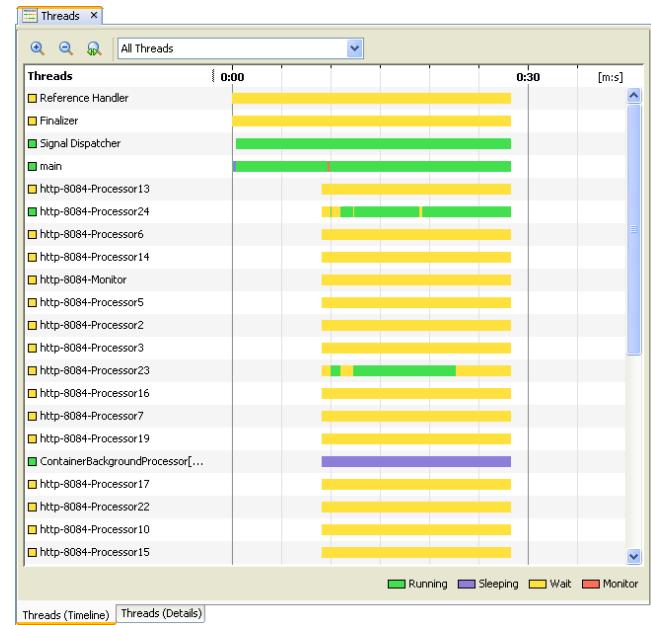
Arquitetura





Monitoramento de Threads

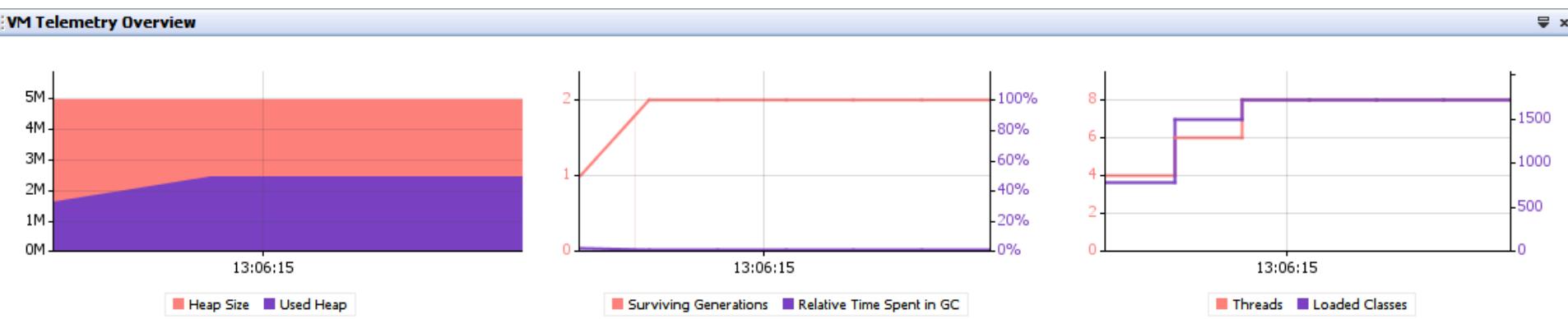
- Linha do tempo de Threads
 - Nome da Thread, classe
 - Status da Thread no tempo
- Detalhes da Thread
 - Estatística de Atividade
 - Timestamp para cada estado
 - Descrição de threads de sistema





Monitoramento de Telemetria da VM

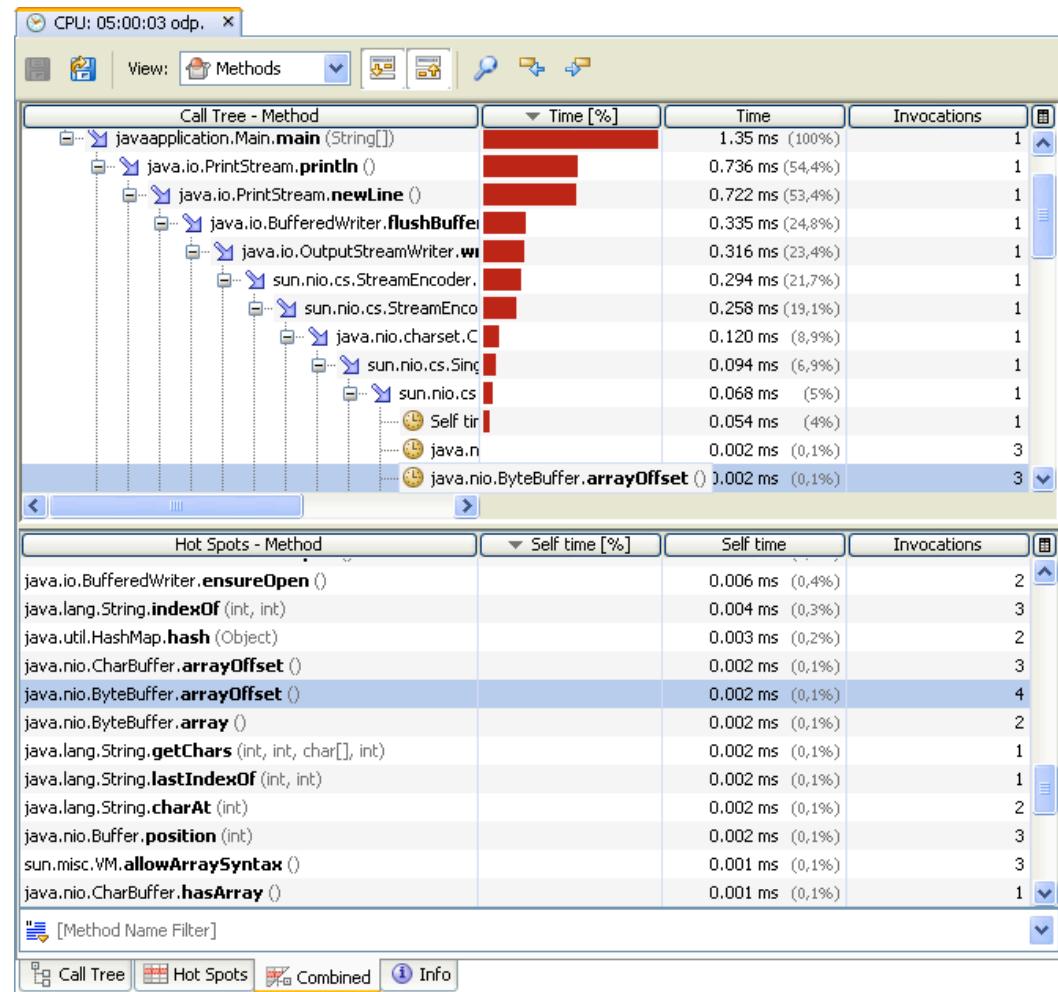
- Visualização da telemetria da máquina virtual
 - Heap alocada, heap usada
 - Número de Threads executando na VM
 - Tempo relativo gasto pelo garbage collection
 - Geração de métricas de detecção de memory leak





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

