Gridify your Spring application with Grid Gain

Sergio Bossa
Pro-Netics / Sourcesense
About me

✔ Software architect and engineer
  ➔ Pro-Netics (http://www.pronetics.it)
  ➔ Sourcesense (http://www.sourcesense.com)

✔ Blogger
  ➔ http://sbtourist.blogspot.com

✔ Open Source Enthusiast
  ➔ Lead at Scarlet - Clustering for Jira (http://scarlet.sf.net)
  ➔ Committer at Spring Modules (http://springmodules.dev.java.net)
  ➔ Committer at Taconite Ajax Framework (http://taconite.sourceforge.net/)
Agenda

- Common ground.
  - Why Grid?
  - Performance and Scalability.
  - Map / Reduce.
- Grid Gain and The Spring Framework.
  - Grid Gain concepts.
  - Grid Gain on Spring.
  - A practical example.
Why grid?
Social changes

- Cheaper hardware.
  - Computer as an off-the-shelf product.
- Web explosion.
  - Everything is on the web.
  - Everyone is on the web.
  - More and more users.
  - More and more transactions.
Why grid?
Technological changes

✔ Cheaper hardware.
  ➔ We have more power.
    ➔ We want more speed.

✔ Moore's Law: the number of transistors that can be inexpensively placed on an integrated circuit is increasing exponentially.
  ➔ But we had to go from single-core processors to multi-core ones ...
  ➔ So your for-loop will always run at the same speed.
    ➔ Forever.
Why grid?
Performance

It's all about doing one thing. Faster.
Scalability

It's all about doing the same one thing.
In a bigger way.
Performance vs Scalability

✓ Performance is about *how fast*.
✓ Scalability is about *how much*.
✓ Nowadays, if you want to save your job and hears (remember that Boss screaming at your face) ... ➔ You have to scale.
Scalability in two words

- **Vertical** scalability is about adding more and more power (CPU, RAM ...) to your single computer.
  - Also known as *scaling-in*.
  - Finite and costly.

- **Horizontal** scalability is about adding more and more computers.
  - Also known as *scaling-out*.
  - Infinite and cheaper, because using commodity hardware.

- Guess what, we want to **scale-out** ...
The scalability factor

- Available resources while scaling out.
  - Linear scalability.
  - Supra-linear scalability.
  - Sub-linear scalability.
  - Negative scalability.

- A scalable application should always strive for (almost) linear scalability.
The scalability problem

- **Amdahl's Law**: performance decreases as number of processors increases once there is even a small percentage of non-parallelizable code.
  - Most of the software is written in a non-parallelizable way.
  - Writing software that scales out is perceived as hard.
Entering Map / Reduce

✔ From Google Labs.

→ Is it enough?
Map / Reduce explained

- Programming model for linearly scaling out.
- De-facto standard for parallelizing intensive processing tasks.
- Based on:
  - Splitting tasks into several parallelizable jobs grouped by key.
  - Mapping jobs to processing units, optionally taking into account the job key.
  - Merging jobs results, joining them into a global task result.
Map / Reduce illustrated
Counting words
Grid computing with Map / Reduce

✔ Grid computing.

→ Basically, a way to exploit multi-core / multi-processors / multi-computer environments for achieving horizontal scalability.

✔ Map / Reduce.

→ Common paradigm in grid computing for implementing scalable applications.
Entering Grid Gain

- Open Source Grid Computing Framework.
  - Web: http://www.gridgain.com
  - Created and supported by Grid Gain Systems.
    - Community support.
    - Professional support.
- Powerful, yet simple, yet fun, Map / Reduce implementation.
- Integrated with major servlet containers and application servers.
- Integrated with major data grid solutions.
- Integrated with the Spring Framework.
Task arrives to the first grid node, where is split into three jobs.
First job is self-assigned and processed.
Second job is sent to the second grid node, where is processed.
Third job is sent to the third grid node, where is processed.
Result from the second job is collected by the task on the first node.
Result from the third job is collected by the task on the first node.
Collected job results, together with the result from the first job, are reduced by the task and returned as a global result.
Grid Gain Quick Start

✔ GridTask.
  ➔ Implements the Map / Reduce logic.

✔ GridJob.
  ➔ Implements the processing logic.

✔ GridFactory.
  ➔ Provides access to the grid for executing tasks.

✔ Automatic deployment.
  ➔ Tasks are automatically deployed to the grid.

✔ Peer class loading.
  ➔ Needed classes are automatically loaded from peers.
Grid Gain Advanced

- SPI (Service Provider Interface) based configuration.
  - Discovery SPI.
  - Topology SPI.
  - Checkpoint SPI.
  - Load Balancing SPI.
  - Collision SPI.
  - Failover SPI.
  - Metrics SPI.
  - ...
Entering Spring Framework

The leading full-stack Java/JEE application framework.
Grid Gain on Spring

- POJO configuration.
- AOP grid execution.
- Resource Look-up.
Spring-based configuration

- POJO-based.
- Spring-based.
- GridConfiguration
  - Configure grid parameters.
  - Configure actual SPI implementations.
  - Declared as a Spring bean.
- GridFactory
  - GridFactory.start(GridConfiguration cfg)
  - GridFactory.start(String springCfg)
AOP-based grid execution

✓ Parallelization on grid as a cross-cutting concern.
✓ Transparent task deployment and execution.
✓ Gridify
  ➔ Annotation to identify methods that must be executed on grid.
✓ GridifySpringEnhancer
  ➔ Proxy-based enhancer for executing annotated object methods on grid as an aspect.
Container-based resource look-up.

✓ Spring application context as a source for resources needed by tasks and jobs.

✓ GridSpringApplicationContextResource
  ➔ Annotation for injecting the Spring application context into tasks and jobs.

✓ GridFactory.start(GridConfiguration cfg, ApplicationContext springCtx)
  ➔ Starts grid with a specific context to use for looking-up resources.
public class WordCounter {

@Gridify(taskClass = WordCounterGridTask.class, gridName = GridStarter.GRID_NAME)
public Map<String, Integer> count(Set<String> fileNames, Set<String> words) {
    Map<String, Integer> result = new HashMap<String, Integer>();
    for (String fileName : fileNames) {
        String fileContent = this.readFile(fileName);
        StringTokenizer fileTokenizer = new StringTokenizer(fileContent, " ,.;\n\r\t");
        while (fileTokenizer.hasMoreTokens()) {
            String token = fileTokenizer.nextToken();
            if (words.contains(token)) {
                Integer wordOccurrence = result.get(token);
                if (wordOccurrence == null) {
                    result.put(token, 1);
                } else {
                    result.put(token, ++wordOccurrence);
                }
            } else {
                result.put(token, ++wordOccurrence);
            }
        }
    }
    return result;
}

// ...
An Example
The Grid Task

```java
public class WordCounterGridTask implements GridTask<GridifyArgument, Map<String, Integer>> {
    @GridLoadBalancerResource private GridLoadBalancer balancer;
    @GridLoggerResource private GridLogger logger;

    public Map<? extends GridJob, GridNode> map(List<GridNode> nodes, GridifyArgument args) throws GridException {
        Map<GridJob, GridNode> mapping = new HashMap<GridJob, GridNode>(nodes.size());
        Set<String> fileNames = (Set<String>) args.getMethodParameters()[0];
        Set<String> words = (Set<String>) args.getMethodParameters()[1];
        for (String fileName : fileNames) {
            WordCounterGridJob job = new WordCounterGridJob(fileName, words);
            mapping.put(job, this.balancer.getBalancedNode(job));
        }
        return mapping;
    }

    public GridJobResultPolicy result(GridJobResult currentResult, List<GridJobResult> processedResults) throws GridException {
        if (currentResult.getException() == null) {
            return GridJobResultPolicy.WAIT;
        } else {
            this.logger.error(currentResult.getException().getMessage(), currentResult.getException());
            throw new IllegalStateException(currentResult.getException());
        }
    }

    public Map<String, Integer> reduce(List<GridJobResult> jobResults) throws GridException {
        Map<String, Integer> globalResult = new HashMap<String, Integer>();
        for (GridJobResult jobResult : jobResults) {
            Map<String, Integer> perJob0currences = jobResult.getData();
            for (String word : perJob0currences.keySet()) {
                Integer global0currence = globalResult.get(word);
                if (global0currence == null) {
                    globalResult.put(word, perJob0currences.get(word));
                } else {
                    globalResult.put(word, global0currence + perJob0currences.get(word));
                }
            }
        }
        return globalResult;
    }
}
```
public class WordCounterGridJob extends GridJobAdapter {

    @GridSpringApplicationContextResource private ApplicationContext springContext;
    @GridLoggerResource private GridLogger logger;
    private WordCounter counter;
    private String fileName;
    private Set<String> words;

    public WordCounterGridJob(String fileName, Set<String> words) {
        this.fileName = fileName;
        this.words = words;
    }

    public Serializable execute() throws GridException {
        this.counter = (WordCounter) this.springContext.getBean("counter");
        Map<String, Integer> result = this.counter.count(new HashSet<String>(Arrays.asList(this.fileName)), this.words);
        this.logger.info("Occurrences found for " + this.fileName + " : " + result);
        return (Serializable) result;
    }
}
An Example
Grid Configuration

```xml
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:util="http://www.springframework.org/schema/util"

  <bean id="gridCfg" class="org.gridgain.grid.GridConfigurationAdapter">
    <property name="gridName">
      <util:constant static-field="com.sourcesense.gridgain.wordcounter.grid.starter.GridStarter.GRID_NAME"/>
    </property>
    <property name="gridGainHome" value="/opt/gridgain-2.0.2"/>
    <property name="checkpointSpi">
      <bean class="org.gridgain.grid.spi.checkpoint.sharedfs.GridSharedFsCheckpointSpi">
        <property name="directoryPath" value="/tmp/gridgain"/>
      </bean>
    </property>
  </bean>

  <bean id="gridifiedCounter" class="org.gridgain.grid.gridify.aop.spring.GridifySpringEnhancer" factory-method="enhance">
    <constructor-arg ref="counter"/>
  </bean>

  <bean id="counter" class="com.sourcesense.gridgain.wordcounter.WordCounter"/>

</beans>
```
public class GridStarter {

    public final static String GRID_NAME = "WordCounterGrid";

    public static void main(String[] args) throws GridException, InterruptedException {
        ApplicationContext gridContext = new ClassPathXmlApplicationContext("classpath:grid-context.xml");
        try {
            GridFactory.start((GridConfiguration) gridContext.getBean("gridCfg"), gridContext);
            Thread.sleep(180000);
        } finally {
            GridFactory.stop(GRID_NAME, false);
        }
    }
}
public class WordCounterGridTest extends AbstractDependencyInjectionSpringContextTests {

    private final static String FILE_1 = "/tmp/test1";
    private final static String FILE_2 = "/tmp/test2";
    private WordCounter gridifiedCounter;
    private GridConfiguration gridCfg;

    public WordCounterGridTest(String testName) {
        super(testName);
        this.setAutowireMode(AUTOWIRE_BY_NAME);
    }

    public void setGridifiedCounter(WordCounter gridifiedCounter) {
        this.gridifiedCounter = gridifiedCounter;
    }

    public void setGridCfg(GridConfiguration gridCfg) {
        this.gridCfg = gridCfg;
    }
}
public void testGridifiedWordCounter() {
    try {
        Map<String, Integer> result = this.gridifiedCounter.count(
            new HashSet<String>(Arrays.asList(FILE_1, FILE_2)),
            new HashSet<String>(Arrays.asList("java", "scala", "groovy")))
        assertEquals(3, result.size());
        assert Equals(new Integer(3), result.get("java"));
        assertEquals(new Integer(3), result.get("scala"));
        assertEquals(new Integer(1), result.get("groovy"));
    } catch (Throwable ex) {
        ex.printStackTrace();
        fail(ex.getMessage());
    }
}

protected void setUp() throws Exception {
    if (GridFactory.getState(GridStarter.GRID_NAME).equals(GridFactoryState.STOPPED)) {
        GridFactory.start(this.gridCfg, this.applicationContext);
    }
}

protected String[] getConfigLocations() {
    return new String[] {"classpath:grid-context.xml"};
}
Thank you

Sergio Bossa
s.bossa@pronetics.it
s.bossa@sourcesense.com