JSR 352 Expert Group

Working Session

21 March 2012

Agenda

- Checkpoint: Annotations vs XML
- Finish up: Parallelization
- Discussion: Job Context
- List for Next Meeting

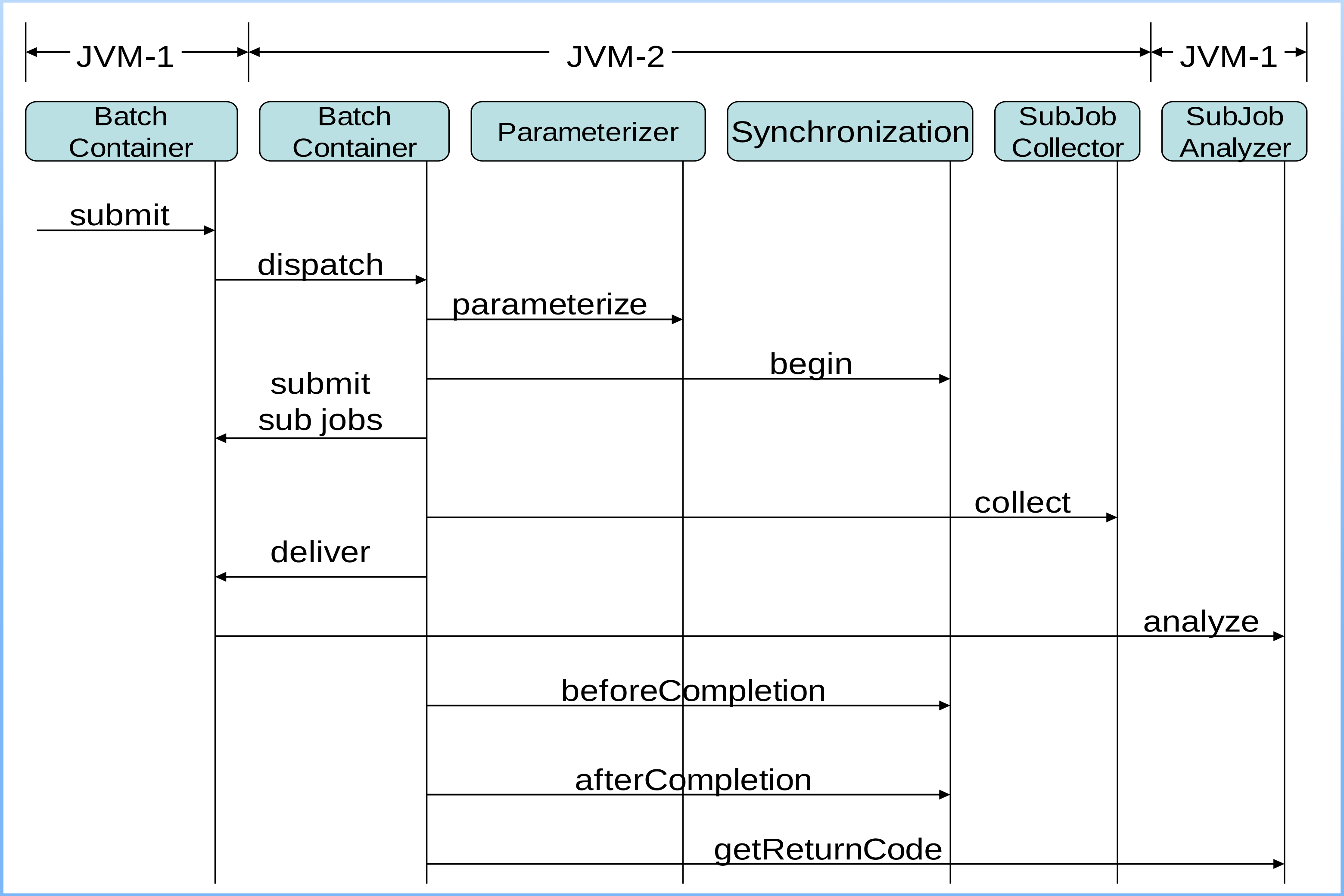
Finish Up: Parallelization

- Reconcile Spring/WebSphere partitioned batch differences
- Spring:
  - <partition step="step1" partitioner="partitioner"> <handler grid-size="10" task-executor="taskExecutor"/> </partition>
  - Grid size known to job and task execution algorithm is configurable at job level.
- WebSphere:
  - <run instances=multiple jvms=single/> <prop name="com.ibm.websphere.batch.parallel.parameterizer" value={Parameterizer impl class}/>
  - Grid size is known to batch container (infrastructure)
  - Task execution algorithm is an extensible part of the batch container
  - Parameterizer (partition algorithm) determines number of partitions and unique parameters for each “sub job” instance
  - Static (XML properties) model for specifying number of partitions and subjob parameters available as alternative to Parameterizer.

Finish Up: Parallelization

- Understanding WebSphere parallelization model
  - Top job/Sub job – sub tasks are jobs
  - Top job and sub jobs support restart
  - Parallel services assignable to job:
    - Parameterizer – partition algorithm, decides number of partitions, job parameters per partition
    - Synchronization – provides logical unit of work demarcation for implementing compensation
    - SubJobCollector – allows one-way communication from sub job to top job (e.g. collect application stats)
    - SubJobAnalyzer – receives information about sub job execution: collector payloads and end state

Finish Up: Parallelization



Discussion: Job Context

- Runtime object that communicates state of current job execution
- Injected by Runtime via annotation
- Holds following information:
  - Job
    - name, parameters
    - End state, return code
    - Metrics
    - Transient and persistent “properties” bags
  - Per step
    - Name, parameters
    - End state, return code
    - Metrics
    - Transient and persistent “properties” bags

Discussion: Job Context

```
package jsr352.example;
import javax.batch.runtime.JobContext;
@ItemProcessor
public class MyItemProcessor {
        @Context JobContext jobCtx;
        @ProcessItem MyOutputItem process(MyInputItem item) {
                // process item
                // update persistent application metric
                Properties p= jobCtx.getStepPersistentProperties();
                int myCount= (int) p.getProperty(“MyCount”));
                if ( <condition>) myCount++;
                p.putProperty(“MyCount”,myCount);
        }
}
```

List for Next Meeting

- Future
  - XML and Bean Instantiation
  - Parameters and XML
  - Exit codes
  - Step conditions
  - Metrics
  - Java EE