VSDB

XVSM Benchmark

Mozartspaces

Daniel Dimitrijevic Thomas Traxler

24. Januar 2014

5AHITT

Inhaltsverzeichnis

1 Aufgabenstellung

Realisieren Sie mithilfe der bereitgestellten Implementierung (Autofabrik) das definierte Koordinationsproblem möglichst effizient mit der XVSM Implementierung "MozartSpaces". Setzen Sie dabei Notifications [2Pkt] und Transaktionen [2Pkt] sinnvoll ein (bei Bedarf auch Aspekte [+2ExtraPkt]).

Die in der Implementierung bereitgestellten Teile des Produktionsroboters sollen in den Space geschrieben werden [2Pkt]. Anschließend sollen die Einzelteile zu Autos zusammengebaut, getestet und schließlich ebenfalls in den Space ausgeliefert werden [2Pkt]. Weitere Anhaltspunkte entnehmen Sie bitte der beigelegten Anforderungsanalyse.

Überlegen Sie sich eine entsprechende Containerstruktur [2Pkt]. Jedes Einzelteil und jedes zusammengebaute Auto soll durch ein eigenes Objekt im Space repräsentiert werden (und nicht etwa in einer Liste, die als Ganzes in den Space geschrieben wird). Jeder Roboter wird dabei als eigener Prozess gestartet. Diese Prozesse können als einfache Konsolenapplikationen implementiert werden, bei denen die ID als Argument übergeben wird. Die Kommunikation zwischen den Robotern darf nur über den gemeinsamen Space erfolgen, der vor dem Starten der Roboter gestartet und evtl. initialisiert werden muss. Die Anzeigetafel erhält alle Informationen nur über den Space [2Pkt]. Prozesse für die Roboter sollen jederzeit dynamisch gestartet oder geschlossen werden können, ohne dass die Funktionalität beeinträchtigt wird.

Stellen Sie zu guter Letzt in einem Benchmark Ihre XVSM-Implementierung der RMI-Implementierung gegenüber und überprüfen Sie auf Performance, Skalierbarkeit und Aufbau des Source-Codes [4Pkt]. Beachten Sie dabei mögliche Patterns. Dokumentieren Sie ausführlich all Ihre Codeänderungen. Vergessen Sie dabei nicht auf die Beschreibung der neu verwendeten Technologien.

2 XVSM

XVSM (eXtensible Virtual Shared Memory). XVSM ist eine Middleware Technology die Daten in “Container” speichert und für andere Peers teilt. Dieser Weg bringt einige Vorteile wie die Daten werden auf mehreren verschiedenen Computern verteilt, damit wird die Ausfallwahrscheinlichkeit der Daten reduziert.

3 Arbeitsaufteilung

Name	Arbeitssegment	Time Estimated	Time Spent
Traxler Dimitrijevic	Protokoll	0.5h	0.5h
Traxler	Container Struktur	1h	1h
Traxler	CarFactory to XVSM	2h	2.5h
Dimitrijevic	Roboter to XVSM	2h	3.5h
Dimitrijevic	Testing	1h	1h
Dimitrijevic Traxler	Benchmarking	1h	1h
Gesamt		7.5h	9.5h

4 UML

5 Arbeitsdurchführung

Ein großes Problem das wir vorfanden war, dass das Mozartspaces System bei jeder Verbindung neue Threads erzeugt welche weiter wartend waren nachdme die Verbindung beendet wurde so sammelte sich eine Unzahl an TCP-Connection Threads an wenn das System zuviele Anfragen bekommen hat.

6 Benchmarks

6.1 Unter RMI

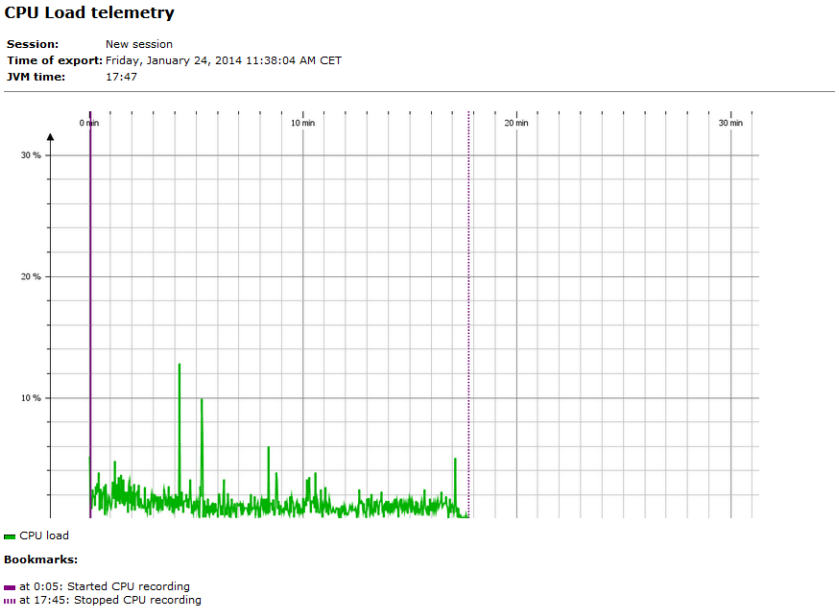

Abbildung 6.1:

6.2 XVSM

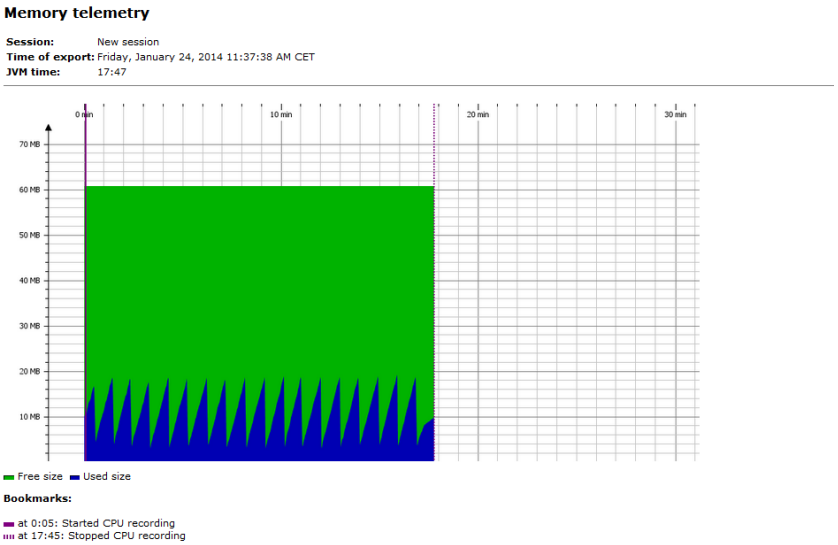

Abbildung 6.2:

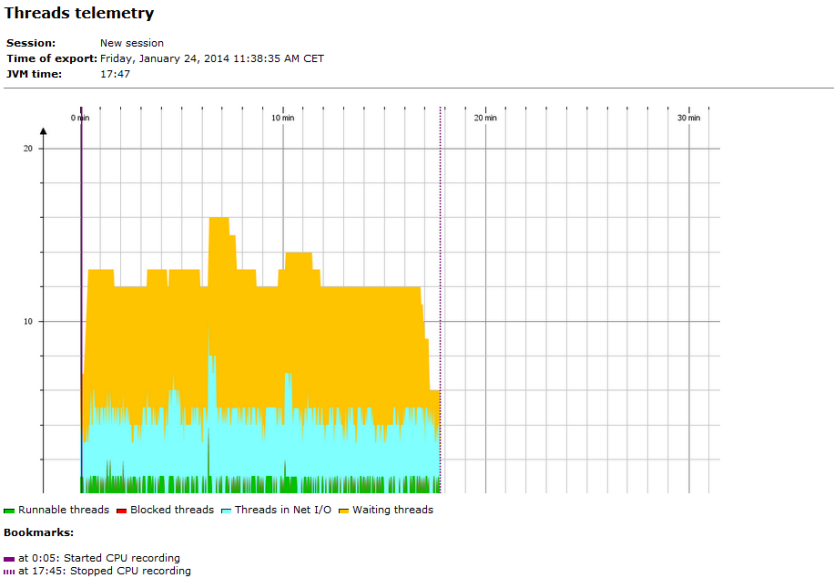

Abbildung 6.3:

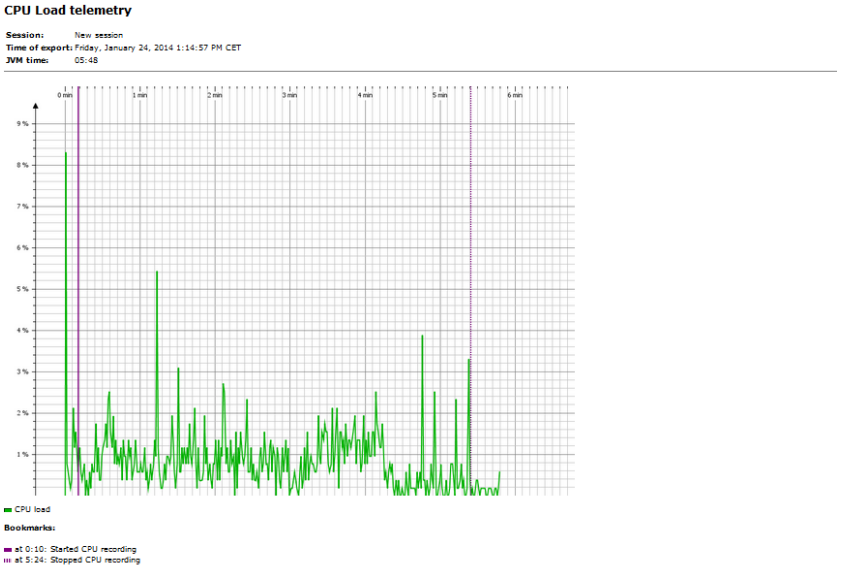

Abbildung 6.4:

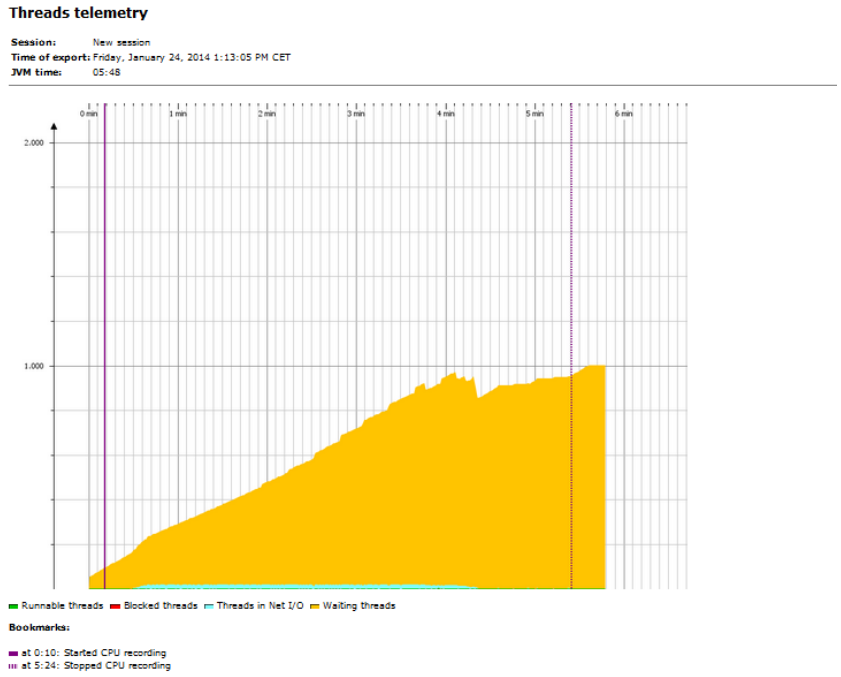

Abbildung 6.5: