Oracle Coherence für Dummies
In-Memory Computing für Einsteiger

Gabriele Jäger
Oracle Deutschland B.V. & Co.KG
Geschäftsstelle Düsseldorf

Schlüsselworte

Oracle Coherence, In-Memory Data-Grid, Cluster, WebLogic Server, Cache

Einleitung

Wer kennt das nicht? Trotz Einsatz aktueller Hard- und Softwarekomponenten, optimaler Datenstrukturen und Algorithmen, Tuning-Maßnahmen u.v.m. zeigt meine Anwendung nach einiger Zeit nicht mehr die erwartete Performanz. Meistens liegen dann die Ursachen bei mangelnder Kapazität und zu hoher Latenz. Die speicherresidente Data-Grid-Lösung Oracle Coherence schafft hier Abhilfe und ermöglicht eine vorhersagbare Skalierung von Anwendungen für einen schnellen und zuverlässigen Zugriff auf häufig benötigte Daten, Datenanalysen in Echtzeit, Berechnungen im In-Memory Grid sowie eine parallele Transaktions- und Ereignisverarbeitung.

Der Vortrag zeigt neben den nachfolgend beschriebenen Grundprinzipien der Oracle In-Memory Data-Grid Lösung auch den Mehrwert durch Anwendungsbeispiele wie RDBMS- und Mainframe-Offload, Anwendungen mit Zustandsinformationen in Webanwendungen, Http-Session Management, Objektinteroperabilität (.NET, Java), u.v.m. auf.

Was ist ein In-Memory Data-Grid?

Der Java Community Weblog Service[1] beschreibt ein Data-Grid wie folgt:

1. Ein Data-Grid ist ein System von mehreren Servern (Knoten), welches Informationen in Form von Anwendungsobjekten auf diesen Knoten verwaltet und Operationen (zum Beispiel Berechnungen) mit diesen Informationen ausführt.
2. Ein In-Memory Data-Grid ist ein Data-Grid, welches Informationen In-Memory speichert um eine sehr hohe Performanz zu erreichen.
3. Informationen werden in Anwendungsobjekten (Domain Objects) gehalten. Kopien der Objekte sind redundant auf verschiedenen Knoten des Systems verteilt. Diese Eigenschaft bildet die Grundlage für die Elastizität des Systems und die Hochverfügbarkeit der Information im Falle eines Knotenausfalls. Die Struktur der Anwendungsobjekte wird mittels objektorientierter Sprachen definiert, wie z.B. Java (POJOs), C++, C#. Es existieren Schnittstellen (API) auf Basis dieser Sprachen, um Create, Read, Update und Delete (CRUD) Operationen, Abfragen und Berechnungen durchzuführen.
4. Ein In-Memory Data-Grid speichert jegliche Informationen im Hauptspeicher und führt dort auch alle Operationen aus. Dies ist die Grundlage für hochperformantes, verteiltes Rechnen, da der Zugriff auf persistent gehaltene Strukturen vermieden wird (Vermeidung von Objekterzeugung aus relationalen Strukturen, Vermeidung von File I/O u.ä.).

5. Anwendungsobjekte sind auf den verschiedenen Knoten des Systems gemeinsam nutzbar. Andere Systeme (z.B. Middleware Anwendungen) können auf diese Objekte transparent zugreifen – d.h. das auf den In-Memory Data-Grid zugreifende System muss nicht wissen, auf welchem Knoten sich die Objekte befinden.
6. Die Objekte können Informationen aus anderen Systemen (RDBMS, File-System, u.a.) „widerspiegeln“ und können, falls notwendig in diese Systeme geschrieben werden oder von dort gelesen werden. Das Schreiben kann synchron als auch asynchron erfolgen.

Was ist Oracle Coherence [2]?

Ein Coherence Knoten ist nichts anderes als ein Java Prozess (JVM Prozess), der mit den geforderten Coherence Java Bibliotheken und gewissen Coherence Konfigurationsinformationen gestartet wurde. Lädt man Oracle Coherence vom Oracle Technology Network OTN herunter, erhält man ein 86 MB großes zip-File, welches im wesentlichen eine Handvoll Java Bibliotheken enthält. Diese haben keine Abhängigkeiten zu Drittpartei Bibliotheken. Es wird nur eine JDK Installation vorausgesetzt.

Ein Coherence Cluster ist eine Menge von solchen Java Prozessen (auf einem Rechner bzw. auf verschiedenen physikalischen Rechnern), mit ähnlicher Konfiguration, die miteinander kommunizieren. Ein Client, geschrieben in Java, C++ oder auch .NET kann sich mit dem Cluster verbinden, um per API gewünschte CRUD (create, read, update, delete) Operationen, Abfragen und Berechnungen auf den Storageknoten (Knoten, auf denen sich die Anwendungsobjekte befinden) ausführen zu lassen.

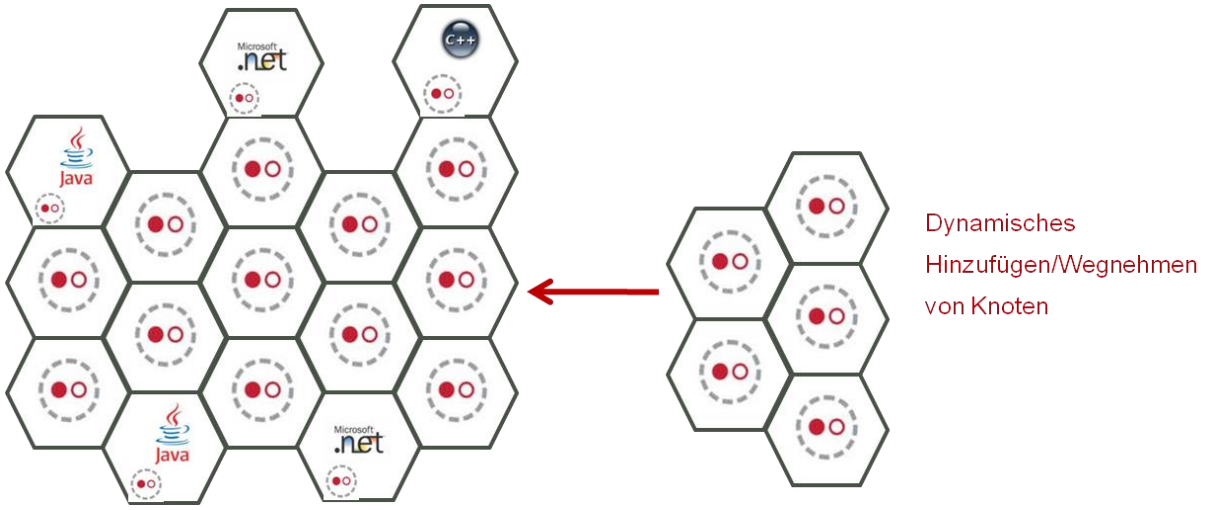

Bild 1: Verschiedene Coherence Knoten formen ein Cluster, Client Knoten (Java, .NET, C++) sind selbst Teil des Clusters

Wird zur Laufzeit zusätzliche Kapazität benötigt, können einfach neue Storage-Knoten gestartet und dem Cluster hinzugefügt werden. Nach Spitzenzeiten können Knoten den Cluster auch verlassen. In beiden Fällen werden die Anwendungsobjekte automatisch im Cluster rebalanciert – also auf den im Cluster befindlichen Knoten gleichmäßig neu aufgeteilt.

Coherence sieht verschiedene Cache Topologien vor, wie zum Beispiel Local Cache, replizierter Cache, partitionierter Cache, Near Cache.

Verschiedene Backing Map Schemes zum eigentlichen Ablegen der Objekte im Cache können zum Einsatz kommen. Je nach Anwendungsfall kann das gewünschte Szenario flexibel umgesetzt werden. Die Konfiguration erfolgt mittels XML und/oder Java System Eigenschaften. Es gilt das Prinzip „Configuration by Default“: nur Verhalten, das vom Default abweicht, muss per Konfiguration überschrieben werden. Für das Anlegen und Manipulieren der XML Konfigurationsdateien gibt es graphische Unterstützung im Oracle Enterprise Pack for Eclipse ab Version 11.1.1.6.

Zudem kann Oracle Coherence Out-of-the-Box für das Management von Http-Session Zuständen in Application Servern mittels Coherence*Web oder für das Ersetzen der Standardcachingmechanismen in verschiedenen Persistenzframeworks (Hibernate L2 Cache, EclipseLink JPA Shared Cache) genutzt werden.

Der Zugriff auf einen Coherence Cache kann per API mittels Java, .NET oder auch C++ geschehen. Das API umfasst unter anderem:

- Zugriffssteuerung auf Cache
- Unterstützung von speziellen, sehr effizienten Serialisierungsmechanismen für die Domain Objekte
- Abfragen durch Filter und Value Extractors
- Erstellen von Indizes
- Benutzen von Aggregatoren
- Benutzen von Entry Processors zur effizienten parallelen Verarbeitung im Cacheverbund
- Definition von Agenten und Benutzen der WorkManager API
- Definition und Abgreifen von Data-Grid Events
- Implementieren von CacheLoader und CacheStore

Oracle Coherence kann mit JMX administriert und überwacht werden. Es existiert eine Integration in den Oracle Enterprise Manager, es stehen aber auch Monitoring Werkzeuge anderer Hersteller zur Verfügung.
Oracle Coherence bietet zudem eine sehr enge Integration mit dem Oracle WebLogic Server und Oracle GlassFish Server, falls Oracle Coherence in Verbindung mit einem dieser Java Applikationserver laufen soll.

Oracle Coherence setzt auf eine komplett geclusterte Architektur auf. Analog eines Telefonkonferenzmodells arbeitet Oracle Coherence mit Coherence Consensus, einem Vertrag zwischen einer Menge von Prozessen über Mitgliedschaft innerhalb des Verbundes zu einer bestimmten Zeit. Dies ermöglicht transparentes, dynamisches und automatisches Failover/Failback von Services und Daten, zuverlässiges Partitionieren von Daten und Services und In-Memory-Funktionalitäten.

Die Software benutzt das Tangosol Coherence Management Protocol (TCMP), ein spezielles proprietäres Peer-to-Peer Unicast-basiertes Protokoll. TCMP ist komplett asynchron, dh. die Kommunikation ist nie blockiert, auch nicht wenn viele Threads eines Servers gleichzeitig kommunizieren und gewährleistet damit eine echte Skalierbarkeit. Darüber hinaus bedeutet die Asynchronität auch, dass die Netzwerk-Latenz nicht den Cluster Throughput beeinflusst, (obwohl es die Geschwindigkeit von bestimmten Operationen beeinflussen kann). TCMP ist ein geclustertes IP-basiertes Protokoll für Server Discovery, Cluster Management, Service Provisioning und Datenübertragung und kann optional Multicast unterstützen.

Oracle Coherence wird vor allem eingesetzt als Puffer für stark frequentierte Daten und zur Reduzierung der Latenz von Backend Systemen. In-Memory Datenabfragen werden 5x bis 20x schneller beantwortet als aus dem Backend.

Warum Oracle Coherence?

Folgende Kerneigenschaften werden mit Oracle Coherence adressiert und schaffen auf der Datenzugriffsebene einer Anwendung Vorteile bezüglich:

- Zuverlässigkeit: das System arbeitet wie geplant, es zeigt ein robustes Verhalten
- Verfügbarkeit: Kostensenkung durch Verminderung von Ausfallzeit, Verbesserung der Benutzerakzeptanz
- Skalierbarkeit: vorhersehbare Skalierbarkeit - bei sich abzuzeichnenden Spitzen kann Kapazität hinzu geschaltet werden. Coherence skaliert linear durch eingebaute Mechanismen (u.a. Clusterarchitektur, partitionierter/Near Cache zur Vermeidung des n*n Synchronisationsproblems, bei der Veränderungen von Daten an einem Knoten eine Synchronisation mit allen anderen Knoten im Verbund nach sich ziehen)
- Ressourcenschonung/Performanz: Verlagerung von Verantwortlichkeiten für Datenhaltung in andere Schicht, mit geringerer Latenz und höherem Durchsatz auf diese Datenzugriffschicht

Der Einsatz einer In-Memory Data-Grid Schicht zieht in der Regel Veränderungen an der Architektur des Gesamtsystems nach sich. Daher sollte er schon beim Entwurf eines Systems berücksichtigt werden.
Wie beim Hausbau gilt auch hier das Motto: Steht das Haus, ist es schwer einen Keller nachträglich einzubauen. In der Realität hat sich jedoch gezeigt, dass Alternativen zu konservativen Architekturen – wie der Einsatz eines In-Memory Grids – auch dann zum Einsatz kommen können, wenn eine der oben stehenden Anforderungen massiv verletzt wird. Ebenfalls sollte für bestimmte Probleme (Skalierbarkeit der Persistenzschicht, Verwaltung von Http Session Zustand) die Nutzung von out-of-the-Box Mechanismen wie z.B. TopLink Grid oder Coherence*Web evaluiert werden, da diese minimal invasiv sind.

Fazit

- Coherence ist eine elegante und praktikable Lösung für Caching
- Coherence ist keine klassische persistierende Datenbank, sondern ein Zwischenspeicher (Cache)
- Es gibt umfangreiche Möglichkeiten, an Coherence anzudocken
- Coherence entlastet das Backend und beschleunigt die Zugriffe
- Dies führt zu weniger Wartezeit durch höhere Kapazitäten

Quellen-Nachweis

[1] http://www.jroller.com/cpurdy/entry/defining_a_data_grid
[2] http://docs.oracle.com/middleware/1212/coherence/index.html

Kontaktadresse:

Gabriele Jäger Oracle Deutschland B.V. & Co.KG Geschäftsstelle Düsseldorf Hamborner Strasse 51 D-40472 Düsseldorf +49 (0) 211-74839 718 gabriele.jaeger@oracle.com www.oracle.com/de	Michael Fuhr Oracle Deutschland B.V.& Co.KG Geschäftsstelle Frankfurt Robert-Bosch Strasse 5 D-63303 Dreieich +49 (0) 6103-397 773 michael.fuhr@oracle.com www.oracle.com/de