

ベンチマークレポート  
ーデータグリッド Caché 編ー

平成 22 年 9 月

グリッド協議会 先端金融テクノロジー研究会 ベンチマーク WG

《 目 次 》

<b>1.</b>	<b>CACHÉ (INTERSYSTEMS)</b> .....	<b>1</b>
1.1	Caché の機能概要 .....	1
1.2	Caché の評価結果 .....	2
1.2.1	ベンチマーク実行環境.....	2
1.2.2	評価シナリオ0:事前テスト.....	3

## 1. Caché (InterSystems)

### 1.1 Caché の機能概要

InterSystems Caché® は、リレーショナル・データベースよりも高速に SQL を実行する、高パフォーマンスなオブジェクトデータベースである。Caché を用いることで、最小のメンテナンスで、迅速な Web アプリケーションの開発、超高速処理速度、驚異のスケラビリティと、トランザクショナルデータに対するリアルタイムクエリが実現可能である。

Caché は、開発者が Web アプリケーションやクライアント/サーバー型アプリケーションを迅速に開発するために必要な機能を提供する。Caché を使用する開発者は、開発ツール、プログラミング言語、データへのアクセス手法などを自由に選ぶことができる。Caché をベースとしたトランザクション処理アプリケーションは、その傑出した性能、高いスケラビリティ、リアルタイム・データ分析、ゆるがぬ信頼性に支えられている。トランザクション処理では性能が重要である。Caché のデータ・サーバー・テクノロジーを使用すると、スピードを損なうことなく最高で数万ユーザーのレベルまでアプリケーションをスケールアップすることが可能である。Caché データサーバの持ついくつかの特徴を以下に示す。

#### (1) 多次元データ・エンジン

データはすべてまばらな多次元配列に格納されており、リレーショナル・データベースで頻繁に発生する「join」操作に関連した処理オーバーヘッドを解消できる。高性能、高スケラビリティ、現実に添った形での複雑なデータのモデリング、データを効率的に格納することによるディスク容量の節約、といった特徴を有する。

#### (2) オブジェクト・データ・アクセス

データはオブジェクトとしてモデル化できる。Caché はカプセル化、多重継承、多態性、埋め込みオブジェクト、参照、コレクション、リレーションシップ、BLOB などをサポートする。迅速なアプリケーション開発、複雑なデータの直感的なモデリングを可能とする。

#### (3) SQL データ・アクセス

Caché データベースにリレーショナルなアクセスが可能。ODBC と JDBC の両方をサポート。レガシーなリレーショナル・アプリケーションの性能を向上させ、標準的な問い合わせ、レポート、分析の各ツールに対する SQL 接続を提供する。

#### (4) 多次元データ・アクセス

Caché データベース中の多次元構造を直接コントロールする。高性能で、レガシー・システムへ接続可能という特徴を有する。

#### (5) 統合データ・アーキテクチャ

単一のデータ定義からオブジェクト・クラスとリレーショナル・テーブルを自動的に生成可能である。迅速な開発、オブジェクトとテーブル間の「インピーダンスミスマッチ」を解消する、という特徴を有する。

#### (6) トランザクション・ビットマップ・インデックス

Caché のビットマップ・インデックスは超高速に更新でき、「生」データとの同時使用に適している。複雑な問い合わせへの高速応答が可能である。また、迅速な更新により、高速トランザクション処理を高性能に保ちつつ、リアルタイム・データの分析が可能である。

## (7) 性能監視用 API

SNMP、WMI をサポートしており、これらを利用して主要な監視ツールと接続可能である。アプリケーションの最適化を支援し、性能仕様を満たす明確な方法を提供する。

## 1.2 Caché の評価結果

### 1.2.1 ベンチマーク実行環境

今回の評価で使用した Caché のバージョンは V2008.2 for x64 redhat である。その他の特別に設定した内容は、表 1-1 に示す通りである。

表 1-1 Caché の設定内容

グローバルバッファ	3,000MB
ルーチンバッファ	128MB
カーネルパラメータ	kernel.shmmax = 3,500,000,000
ジャーナル	OFF (I/O の負荷を減らすため)

また、データオブジェクトは図 1-1 のように定義した。

```

Class Grid.DataObject Extends (%Persistent)
{
Property k As %Integer;           // キー
Property data As %Stream.GlobalBinary; // データ (ストリーム形式)

Index keyIdx On k [ IdKey, Unique ];
}

```

図 1-1 Caché におけるデータオブジェクトの定義

ベンチマークを実行するシステムの環境は二通り用意した。一つ目は、図 1-2 に示すデータノード一台によるシンプルな構成である。二つ目は、図 1-3 に示す AP サーバ2台と DB サーバ1台による構成である。

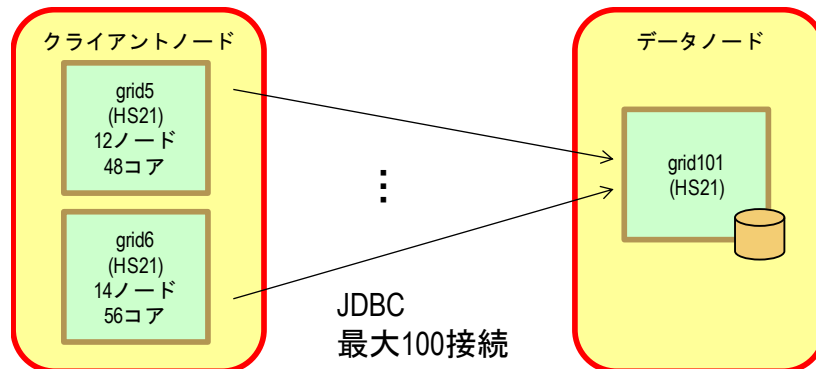


図 1-2 データノード1台構成

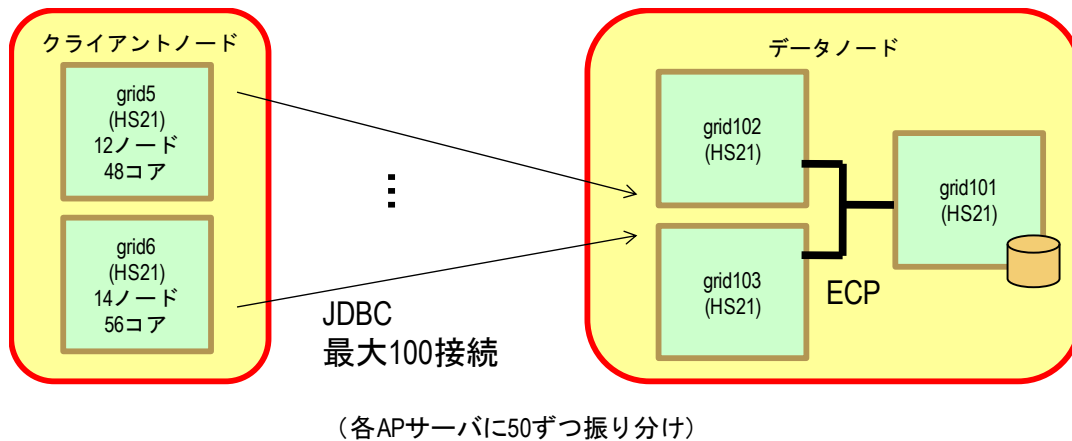


図 1-3 AP サーバ2台と DB サーバ1台による構成

評価シナリオとしては、エラー! 参照元が見つかりません。エラー! 参照元が見つかりません。に述べた内容の一部を実施した。

Caché は、ディスク上のデータのキャッシュをメモリ上に持ち、変更されたデータはディスク上に反映することが基本である。そのため、その他のデータグリッド実装ソリューションとは、測定内容が異なる。

### 1.2.2 評価シナリオ 0 : 事前テスト

評価シナリオ実行に先立ち、ディスク上に有するデータをメモリ上にロードする場合の実行時間について測定を行った。170~250MB/秒の速度が出ている。レコードサイズが大きい場合には、ほぼディスクのアクセス性能に応じた時間でロードできている。レコードサイズが小さく、件数が多い場合には、メモリへのロードにかかる部分が現れてきている。

表 1-2 Caché におけるデータのロード時間

レコードサイズ	レコード数	総データ量	ロードの所要時間
1KB	10 万レコード	100MB	0.582 秒
1MB	1,000 レコード	1GB	4.00 秒

一つ目のテストは、データノード 1 の構成で、オブジェクトサイズを 1KB に固定し、クライアントを 1, 10, 100 と変化させた場合を測定した。アクセスは Read、Update、Write の三通り実行し、その測定結果を図 1-4 から図 1-6 に示す。横軸はクライアント数、縦軸は 1000 個のデータにアクセスするのにかかった時間(ミリ秒)である。

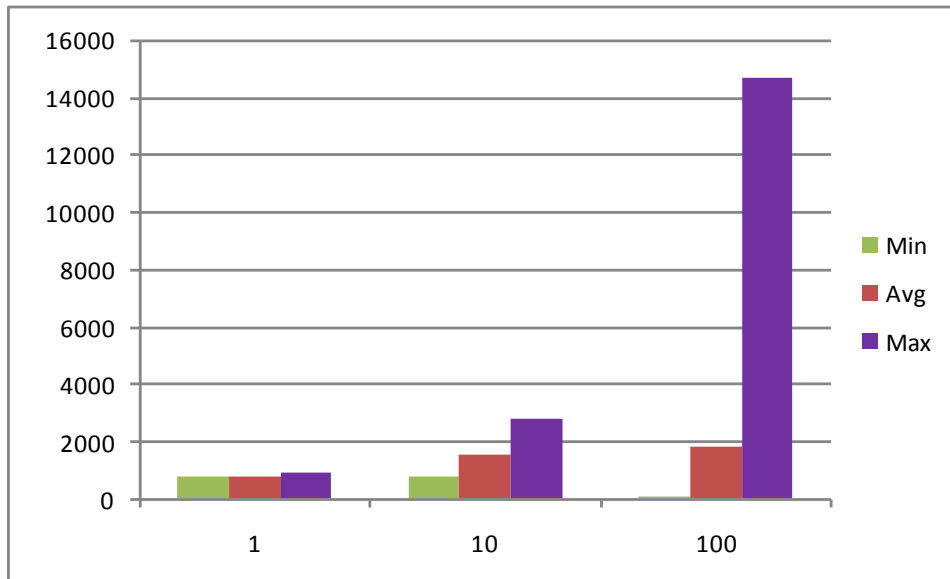


図 1-4 構成1オブジェクトサイズ 1KB、クライアント数を変化(Read)

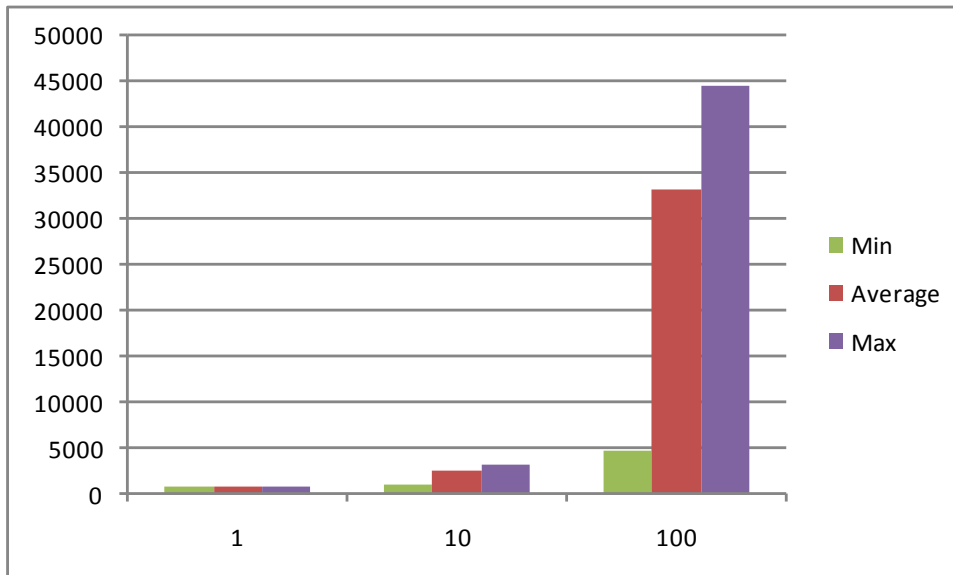


図 1-5 構成1オブジェクトサイズ 1KB、クライアント数を変化(Update)

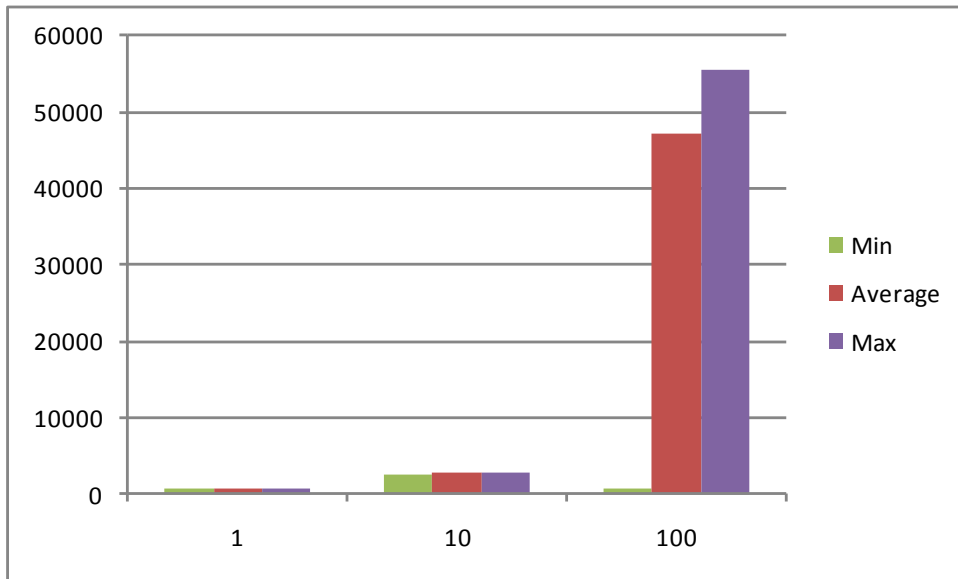


図 1-6 構成1オブジェクトサイズ 1KB、クライアント数を変化(Write)

二つ目のテストは、データノード 1 と 2 の構成で、クライアント数を 10 に固定し、オブジェクトサイズを 1KB, 100KB, 1MB と変化させた場合を測定した。構成1でアクセスを Read, Update, Write の三通り実行した測定結果を図 1-7から図 1-9に示す。横軸はオブジェクトサイズ数(KB)、縦軸は100個のデータにアクセスするのにかかった時間(ミリ秒)である。

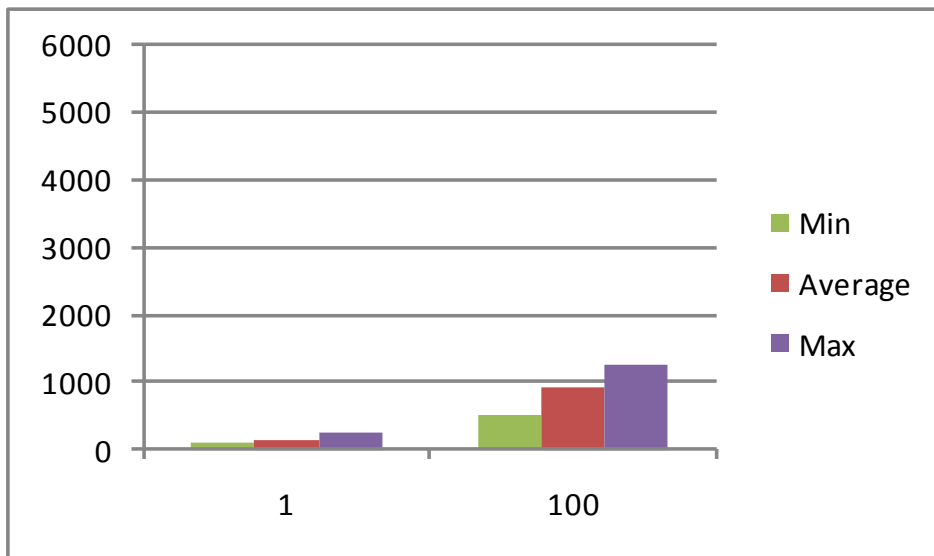


図 1-7 構成1クライアント数 10、オブジェクトサイズを変化(Read)

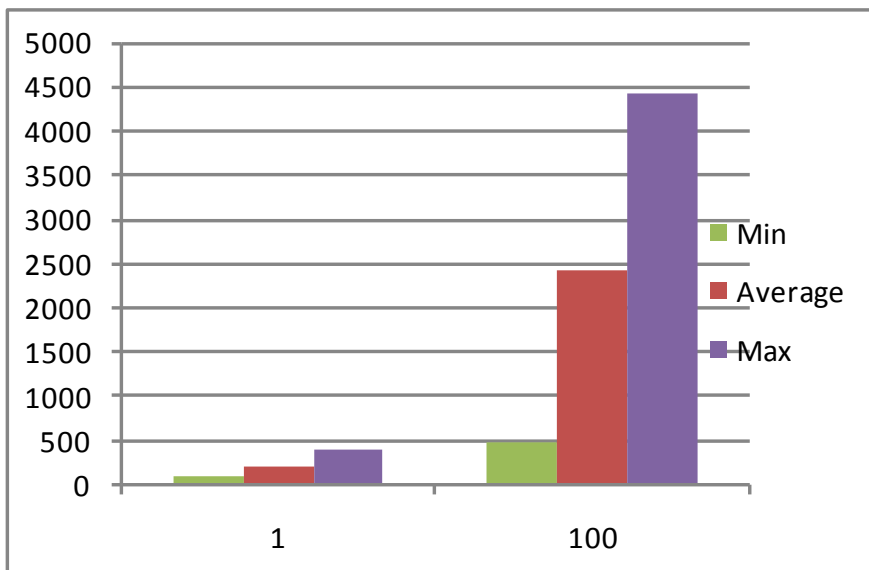


図 1-8 構成1クライアント数 10、オブジェクトサイズを変化(Update)

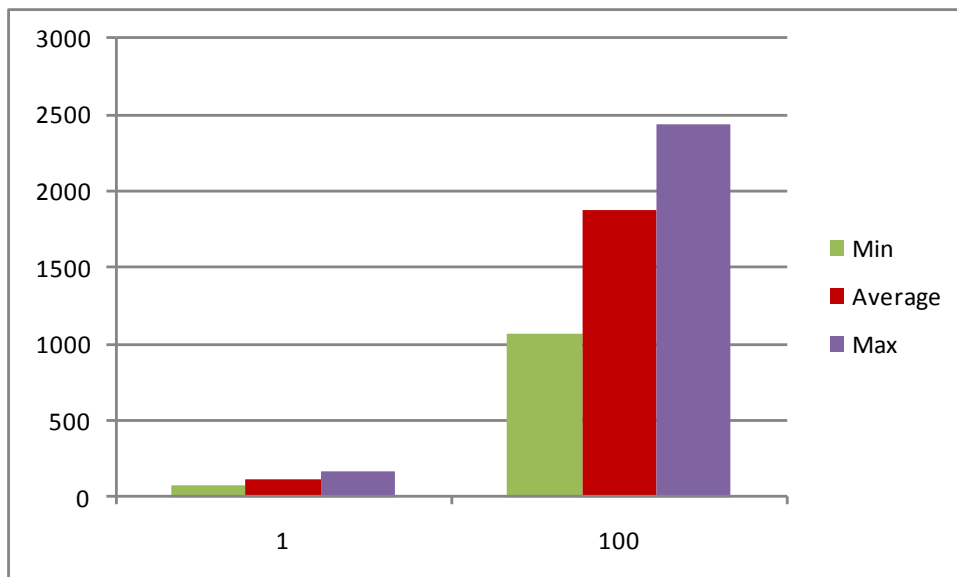


図 1-9 構成1クライアント数 10、オブジェクトサイズを変化(Write)

同様に、構成2でアクセスを Read、Update の三通り実行した測定結果を図 1-7 から図 1-9 に示す。横軸はオブジェクトサイズ数(KB)、縦軸は 100 個のデータにアクセスするのにかかった時間(ミリ秒)である。構成2での Write の測定は、時間の関係で測定できなかった。

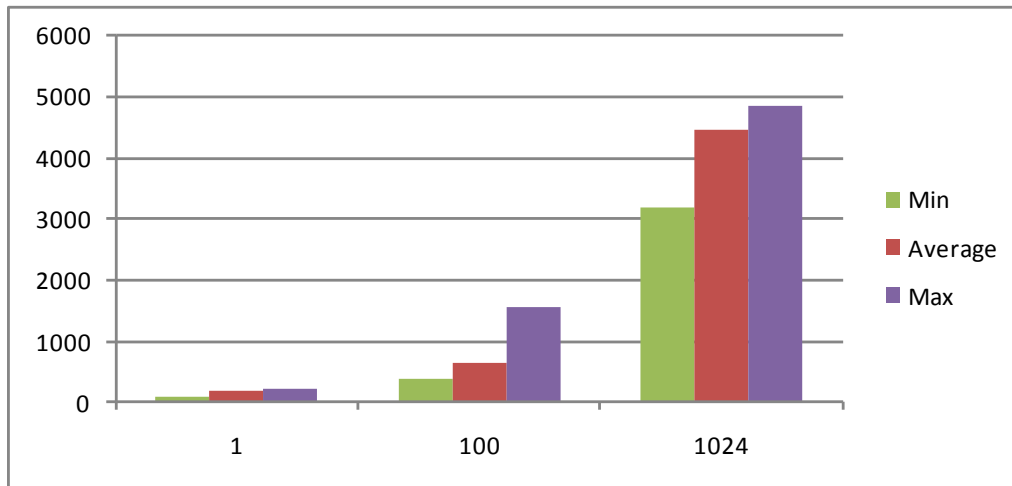


図 1-10 構成2クライアント数 10、オブジェクトサイズを変化(Read)

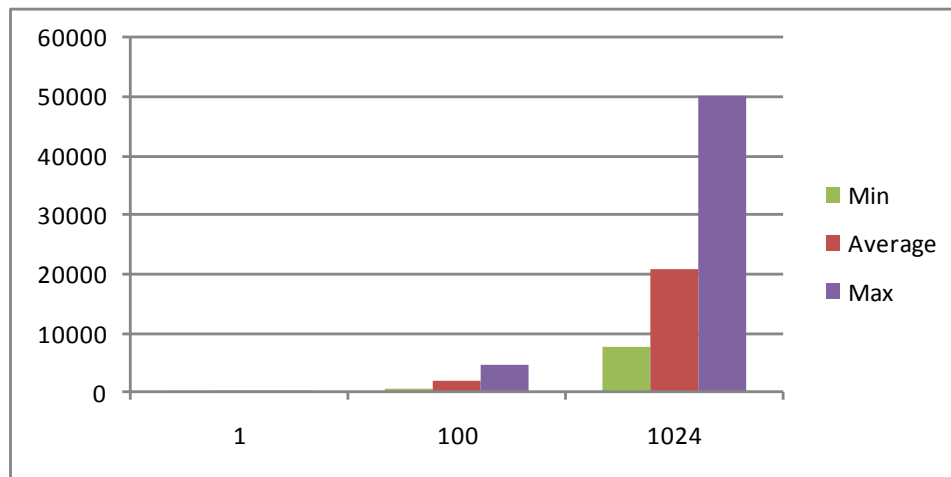


図 1-11 構成2クライアント数 10、オブジェクトサイズを変化(Update)

今回のベンチマークでは、以下の所見を得た。

- AP サーバ 2 台構成では、read のスケーラビリティが確認できた。さらに AP サーバを増やしてもスケーラビリティが得られると期待
- 書き込み操作は、I/O がボトルネックとなる
- ストレージの性能や、データのパーティショニングには検討の余地がある
- vmstat によると、最悪時は%iowait: 約 45%、%idle: 約 50% であり、Disk I/O が大きなボトルネックである
- ロックによる競合の可能性もあるが、判断するに足る情報は今回収集できなかった
- JDBC モジュールのオーバーヘッドが結果に影響を与えた可能性がある。JNI 経由で Caché にアクセスするより高速なインターフェースによるベンチマークを今後の課題としたい