Grid 環境におけるモニタリング手法の評価

秋山智宏[†] 中田秀基[††,†] 松岡聡[†,†††]

広域ネットワーク上に分散した計算資源や情報資源を活用し、大規模計算を行うための Grid と呼ばれる広域分散システムが注目されている。このような環境においては、故障検出、性能予測等のため、システム上の各資源の性能の計測が重要となる。Global Grid Forum 内の組織の一つである Grid Performance Working Group が、モニタリングシステムの基本的なアーキテクチャと XML によるデータ形式を定義・提案しているが、この提案に対しては、1) アーキテクチャのスケーラビリティ、2)XML を用いたデータ表現のコスト、3) データ形式の拡張性、が検証されていない。本研究ではこれらを検証するために、GridRPC システムである Ninf 上に提案アーキテクチャの一部を実装し、評価をおこなった。その結果、アーキテクチャが現実的な設定範囲内では十分スケーラブルであり、XML を用いたデータ形式のコストは許容できる大きさであり、データ形式の拡張性も十分であることを確認した。

Evaluation of Monitoring Method in the Grid

Tomohiro Akiyama ,[†] Hidemoto Nakada [††,†]
and Satoshi Matsuoka[†,†††]

The Grid allows distributed resources to be coordinated in order to facilitate large-scale computing over the wide-area network. In such an environemnt, fault detection and performance monitoring as well as its predeiction becomes one of the important features that need to be agreed upon and possibly standardized. The Grid Performance Working Group within the Global Grid Forum has recently proposed and defined the basic architecture of Grid monitoring and the XML-based data format definitions, but the proposal has been yet tested in practice. In particular, technical concerns include 1) scalability of the proposed architecture, 2) the cost of XML representation of instrumentation events, and 3) extensibility and flexibility of the data definition schema. Our experimental implementation of the part of the proposed architecture on our Ninf GridRPC system has shown that, within a realistic Grid setting the architecture seems reasonably scalable, the added cost of data representation is within permissible bounds, and the schema is sufficiently extensible to accomodidate the specifics of the Ninf system.

1. はじめに

ネットワーク技術の発展により、広域ネットワーク上に分散した計算資源や情報資源を活用して大規模計算を実現する、グローバルコンピューティングが可能となっている。近年、これを目的とした Grid と呼ばれる高性能広域計算システムが複数開発されている[3)4)5)]。

このような Grid 環境においては、故障検出、性能解析、チューニング、及び性能予測、スケジューリング等のための、システム上の各資源の計測が重要である。これを目的とした Grid Monitoring System は複数開発されているが、モニタリングのデータ形式やプロトコル等の相互運用が十分に考慮されているとは言いがたい。

そこで Global Grid Forum(GGF)[6)] 内の組織の一つである Grid Performance Working Group[7)]☆では、モニタリングシステムの基本的なアーキテクチャとその構成要素の相互関係及び XML によるデータフォーマットを定義・提案[8)9)] しており、このようなシステムの相互運用を支援している。

しかしながら、GGF による提案はその有効及び実現可能性、とりわけ、1) モニタリングシステムのス

† 東京工業大学 Tokyo Institute of Technology
†† 産業技術総合研究所 National Institute of Advanced Industrial Science and Technology
††† 科学技術振興事業団 Japan Science and Technology Corporation

☆ Grid 環境におけるリソースやアプリケーションに関する性能情報の収集、表現方法、ストレージ、ディストリビューション、等の基準や最良事例を定義することに焦点を当てており、ツール間における性能データの統一的なフォーマットを開発することと、Grid 環境におけるリソースやサービスをモニタする現存もしくは提案されているツールを分類することが目的である。

ケーラビリティ、2) XML を用いたデータ表現のコスト、 3) データ表現の拡張性、に関して何ら検証が行われていないの実状である。

そこで本研究では、Grid Performance Woriking Group によって定義されたモニタリング手法が Grid 環境において、1) 要求されるモニタリングデータを表現できるか、2) グローバルコンピューティングを行うシステム自体に与える影響が許容できるか、を検証するために、東工大・産総研・RWCP で開発されている Grid RPC システム Ninf[10)11)] に上記の定義に従ったシステムの一部を実装する。さらにこのモニタリングシステムで実験をし、実行プログラムやネットワークに与える影響、また得られたモニタリング情報を処理するコストについて、他の一般的なデータ形式と比較し評価をおこなった。その結果として、アーキテクチャが現実的な設定範囲内では十分スケーラブルであり、XML を用いたデータ形式のコストは大きいものであったがモニタリングシステムが運用される状況では許容できる範囲にあり、データ形式の拡張性も十分であることが確認できた。

2. Global Grid Forum の Performance Working Group の提案の概略

Grid Performance Working Group による Grid Monitoring System に関する提案は、Grid モニタリングシステムの一般的なアーキテクチャと、XML を用いたデータ表現とプロトコルに分けられる。

2.1 Grid Monitoring Service Architecture(GMA)

GMA は以下の 3 つのコンポーネントから成る (図 1 参照)。ここではモニタリングデータを event と呼ぶことにする。

consumer consumer は producer から event を受け取る何らかのプログラムである。これには、実時間でモニタリングデータを集め、得られたデータの解析、アーカイブを行うプログラムや、サーバのプロセスをモニタリングするプログラムなどがある。

producer producer は consumer に対して、event を供給し、directory service に対しては、利用可能な event の情報を公表する。event producer は consumer からの要求に対し“ streaming ”もしくは“ query ”でサービスを提供することが可能である。streaming モードでは、consumer が event 通信路を開き、event が ストリームで返される。一方 query モードでは、consumer は event 通信路を開くことをせず、最も新しい event を要求するだけである。

また、event producer は event に対するアクセスコントロールを提供するためにも用いられる。

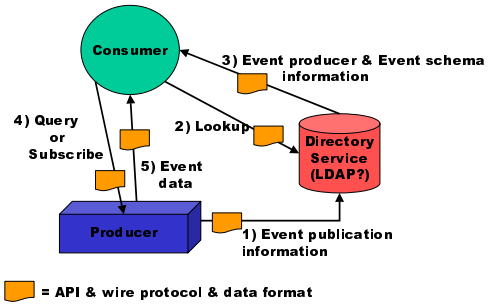

図 1 Gird Monitoring Architecture Components

Grid 環境では基本的にモニタリングされてるリソースを複数の組織が管理している場合が多いので、異なるアクセスポリシー (firewall 等) や、モニタリングする頻度の違い等があると考えられる。いくつかのサイトでは、リアルタイム event ストリームに対しては内部からのアクセスしか受け付けず、外側からは要約したデータしか利用できないかもしれない。producer はこれらのことを厳守しなければならない。

directory service directory service の主な目的は、consumer (users, visualization tool,programs, resource schedulers) に利用可能な情報の検索サービスを提供することである。また、producer 側の情報を更新することができなければならない。directory service は event とそれに関連した producer の一覧のリストを保持しており、consumer が現在利用可能な event の種類、特徴、そしてそれを提供する producer を検索できる。

2.2 Event Schema

モニタリングしたデータ (event) について、何らかの統一された基準に従ってデータ表現することにより、event を異なるツール間で利用することが可能となると考えられる。ここでは、その基準となるスキーマを示す。これは XML を用いたテキスト表現であり、基本的な部分だけが定義されている。これを XSD で拡張することで様々な event を表現することが可能となる。

event は、1 つもしくはそれ以上のデータとしての値をもつ。例えば、システムのロード状況、ネットワークバンド幅、HTTP サーバの状態、アプリケーションの結果などである。

event はひとつの type と複数の element から構成される。

Type event がどんな情報を含んだ集合かを識別するもの。event 内の情報は event type のスキーマに従って構成されている。 つまり、event type

```
<xsd:schema xmlns:xsd=
"http://www.w3.org/1999/XMLSchema"
            xmlns:gp=
"http://www.gridforum.org/PerfWG/schemas">
 <xsd:complexType name="GridEvent"/>
 <xsd:element name="source" type="xsd:string"/>
 <xsd:element name="time" type="xsd:timeInstant"/>
 <xsd:element name=
"sequenceNumber" type="xsd:integer" minOccurs="0"/>
 </xsd:complexType>
</xsd:schema>
```

図 2 base type

はどんな element がその event type 内に必須もしくは任意で必要かを定義する。

Element element は name(string) と value(int, long, float, ... or string) から構成され、単位 (unit) や精度 (accuracy) は含まない。また、必須の element として、Type, Source, Time がある。

2.2.1 XML によるデータ表現

上記で示されていた概念を XML Schema を用いて表現する。XML を用いる理由として以下の 3 つが挙げられる。

(1) XML の文法は単純で、新しいデータ構造を表現するのに適している。

(2) 現時点で XML での記述は商業的に盛んであり、XML ベースのデータを生成、使用するための質の高いツールが利用可能である。

(3) XML Schema のデータベースは生成するのが容易で、わずかな操作で検索したりすることが可能である。

2.2.2 Event Types

Base type は図 2 のように定義されている。2.2 節で定義した必須の Type, Time, Source を element として含んでいる。

個々の具体的な event を表現するためには、この XML スキーマを XSD 記法で拡張する。それぞれの element のタグ内部の構造は type によって違う。例としてロードアベレージを報告する event を考えてみる。一定時間あたり (1、5、15 minutes) の CPU のロードアベレージをモニタしたものを CPULoad Event とする。この GridEvent を拡張したスキーマは図 3 のように記述できる。この event は event type を識別するタグ (<CPULoadEvent>) で始まり、内部には event のソースを記述する XML の要素や生成されたデータが入っている。このスキーマで表現されたロードアベレージの XML 表現を図 4 に示す。

```
<xsd:schema xmlns:xsd=
"http://www.w3.org/1999/XMLSchema"
            xmlns:gperf=
"http://www.gridforum.org/PerfWG/schemas">
 <xsd:annotation><xsd:documentation>
  This event describes the CPU load
           on a host at the specified the time
 </xsd:documentation></xsd:annotation>
 <xsd:element name ="CPULoadEvent">
  <xsd:complexType base=
            "grid:GridEvent" derivedBy="extension"
  <xsd:element name="hostname" type="xsd:string"/>
  <xsd:element name="load1" type="xsd:float"/>
  <xsd:element name="load5" type="xsd:float"/>
  <xsd:element name="load15" type="xsd:float"/>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

図 3 CPULoadEvent Schema

```
<CPULoadEvent>
 <description>1, 5, and 15 minutes
              CPU load average</description>
 <sourcex-event:
        //myhost.gridforum.org:1234</source>
 <time>2001-7-27T13:00-5:00</time>
 <hostName>foo.gridforum.org</hostName>
 <load1>1.2</load1>
 <load5>1.4</load5$>
 <load15>0.9</load15>
<\CPULoadEvent>
```

図 4 CPULoadEvent

3. 問 題 点

ここでは、2 節で述べた手法において想定される問題点を考察する。

アーキテクチャの拡張性 Grid 環境はユーザに対してローカルな環境下にあるクラスタのみからなる場合や、広域分散環境の場合もある。アーキテクチャはこれらの環境の違いに対応できなければならない。

XML を用いたデータ表現のコスト XML で表現した場合、データサイズとパーズコストは通常よりも大きくなる傾向がある。そのため、データサイズが大きくなった場合はネットワークに関して、また パーズコストが大きくなった場合には CPU 性能に関して、 probe effect が生じる可能性がある。ここでいう probe effect とは、モニタリングをする際のデータ転送で生じるネットワークへの負荷や誤差、またパージングによる CPU への負荷や誤差が生じて、本来のモニタリング情報と異なることをいう。

データ表現の拡張性 これは 2.2 節の定義で Grid 環境おけるモニタリングデータを表現できるかということである。

これらの問題に対して、Grid Performance Working Group の定義したモデルに従ったモニタリングシス

テム及び、モニタリングデータのリファレンス実装を行い、検証をする。

4. Ninf 上へのリファレンス実装

以下では、Grid Performance Working Group の定義した GMA と Grid Performance Event に従って、我々がおこなった Ninf 上へのリファレンス実装について述べる。

4.1 Ninf の概要

Ninf は Grid RPC システムの一つであり、クライアントにサーバ側にあるライブラリをリモートから実行する機能を提供する。で、リモートライブラリの呼び出しのことを Ninf_call と呼ぶ。

4.2 Architecture の実装

リファレンス実装したモニタリングシステムは以下の 2 つのコンポーネントからなる。

event writer サーバ上で計算が行われる際に、そのサーバの状態に関する event を収集し、event reader に送信する。

event reader event writer から、event を受信し、パーズして出力をする。

4.3 Grid Performance Event の実装

以下では Ninf システムに特化した 2 つの Grid Performance Event である、NsInvocationEvent と ServerEvent について述べる。

4.3.1 NsInvocationEvent

Ninf_call が一回呼ばれるたびに発生する event を NsInvocationEvent として定義して、以下のような element を持つ。

- ユーザ ID
- Ninf クライアント
- Ninf サーバのホスト名とポート番号
- クライアントからサーバへの通信時間
- サーバにおける実行時間
- クライアントにおける実行時間
- サーバからクライアントへの通信時間
- 実行開始時間
- 実行関数名
- クライアントからの送信データサイズ
- サーバからの受信データサイズ
- プログラムの進行状況

現在の実装では、client と computing steps についてはモニタリングすることはできない。しかし、このデータの必要性と将来的にはこのデータもモニタリング可能になることを考慮してリストに加えた。

図 5 にこの XML 表現である NsInvocationEvent を示す。ここでモニタリングできない要素については、その型が string である場合には値を (null) とし、double,long 等は -1 とした。

```
<NsInvocationEvent>
 <description>(null)</description>
 <source>(null)</source>
 <time>2001-01-11T05:32:41.666-05:00</time>
 <client>(null)</client>
 <usrId>0</usrId>
 <server>assam:3000</server>
 <sendDuration>0.000089</sendDuration>
 <execDuration>0.000137</execDuration>
 <clientExecDuration>0.000154</clientExecDuration>
 <recvDuration>0.000023</recvDuration>
 <startTime>2001-01-11T05:23:41.340-5:00</startTime>
 <functionName>(null)</functionName>
 <sendDataSize>144</sendDataSize>
 <recvDataSize>72</recvDataSize>
 <computingSteps>-1</computingSteps>
<NsInvocationEvent>
```

図 5 NsInvocationEvent

CPU	Cerelon 500MHz
Cache	128KB
Memory	384MB

表 1 Presto クラスタの 1 ノードの性能

4.3.2 ServerEvent

計算資源であるサーバに関しては以下の情報が必要となる。

- サーバホスト名
- サーバとクライアント間のレイテンシ
- サーバクライアント間のスループット
- サーバの空きメモリ容量
- サーバの空きスワップメモリー容量
- サーバ上のプロセス数
- サーバの CPU load
- モニタリング実行間隔

5. 評　　価

評価用のプログラムとして、1)XML を用いたデータ表現のコストを計測するための Micro Benchmark プログラム、2) モニタリングがシステムやネットワークに与える影響を計測するためのモニタリングプログラムを用意した。

5.1 実 験 環 境

評価には東工大松岡研究室の Presto PC クラスタを Ninf サーバとして実験を行った。Presto クラスタの 1 ノードの性能は表 1 の通りであり、64 ノードが 100base-TX のイーサネットで接続されている。以下の実験ではこのクラスタの 16 ノードを用いた。Ninf クライアントとしては、ローカルな環境における実験にホスト A を、グローバルな環境における実験に京都大学工学部建築学科のホスト B を用いた。Ninf サーバとは前者が 100Mbps の LAN で接続されており、後者は 2.4Mbps から 2.7Mbps の WAN で接続されてい

```
(null) (null) 979887469 201990 CLIENT 12345
SERVER 9000 0.000089 0.000137 0.000154
0.000023 979887469 191907 sampl/mmul 144 72 -1
```

図 6 SEQ によるデータ表現

CPU	Pentium III 500MHz × 2
Cache	512KB
Memory	512MB

表 2 評価用計算機

表現方式	データサイズ (byte/event)	parse cost (ms/event)
XML	417	10.3741
SEQ	107	0.2103
XDR	132	0.0441

表 3 Micro Benchmark の結果

る (東工大-京大間)。

5.2 代替データ表現手法

3 節で述べた通り、XML 形式による Grid Performance Event の表現のコストは小さくない可能性がある。そこで、我々は比較のために、1) テキスト形式 (ここでは SEQ 形式と表記する)、2)XDR 形式、による等価なモニタリングデータ表現も実装した。

SEQ 形式はモニタリング情報をテキストで羅列したものである。取得できる情報が図 5 に示されているものであった場合、必須のものであるものも含めると図 6 となる。それぞれの値はスペースで区切られている。XDR(External Data Representation) は機種非依存のバイナリフォーマットである。

5.3 実験結果

5.3.1 Micro Benchmark

ここでは XML、XDR、SEQ 形式で表現された、それぞれの event の単位あたりのデータサイズとそのパーズ処理のコストを計測する。計測する event には、ServerEvent を用い、表 2 の計算機を用いた。また、パーズを行う処理系の実装には Java(Sun JDK 1.3 HotSpotVM) を用いた。

結果を表 3 に示す。各値は 1 万回計測し、平均値をとったものである。この結果から、XML 形式においてはデータサイズ、パーズコスト共に他のデータ表現形式よりも大きいことがわかる。データサイズにおいては他の 2 つよりもおよそ 3〜4 倍であり、パーズコストに至っては XML 形式は、XDR 形式と比較しておよそ 235 倍であり、SEQ 形式と比較しておよそ 50 倍のコストがかかっている。

5.3.2 実環境でのモニタリング

ここでは、実環境におけるモニタリングによるネットワークやシステムへの影響を評価するために、複数の点で event を人工的に発生させ、それを一ヶ所で集計する実験を行う。イベントは、Grid RPC システムで、リモートライブラリ呼び出しにともなってサーバ側で生じるログ情報のイベントとする。サーバとクライアントの間にはライブラリ呼びだしのためにデータ通信が生じるが、その通信速度へのイベント集計による影響を測定することで、モニタリングによるシステムへの影響を評価する。

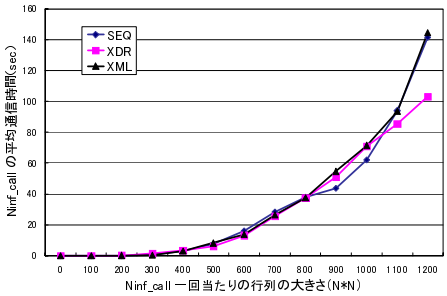

図 7 local 環境における呼び出し間隔一定のモニタリング

event は Ninf のサーバ側ライブラリで発生させる。ここでは、event の集計ホストは、リモートライブラリ呼び出しのクライアントと同一ホストとしている。クライアントは複数のサーバに対して、それぞれ一定の平均間隔をもつポワソン分布に従った間隔でリモート呼び出しを行い、event を発生させる。この際に、特定の大きさの行列をクライアント側からサーバ側へ送信する。サーバ側では任意の仮想的計算時間だけスリープし、その後、最初にクライアント側から受け取ったデータをそのまま返す。その際、サーバ側で event を生成し、event reader に送信する。event reader は受け取ったデータ形式にしたがってデータをパーズし、出力する。

計測する値は Ninf クライアントから Ninf サーバへのリモートライブラリの呼び出し時のデータ転送信時間をサーバ側で計測した値である。これを、送受信する行列のデータサイズと呼び出しを行う間隔を変化させて計測した。

まず、呼び出し間隔を一定にし、送受信するデータサイズを増減して計測を行った。呼び出し間隔の平均値はローカルでは 1 秒、グローバルでは 3 秒となっている。図 7 にローカル環境、図 8 にグローバル環境での結果を示す。

次に、データサイズを一定にし、呼び出し間隔を増減させて計測を行った。データサイズはローカルでは 4M バイト、グローバルでは 40K バイトとなっている。図 9 にローカル環境、図 10 にグローバル環境での結果を示す。

データが大きくなるにつれて Ninf_call の通信時間も大きくなるが、データ形式の違いによる差はそれほど大きなものではないことがわかる。また Ninf_call の間隔を大きくすると、Ninf_call の通信時間が減少するが、これもデータ形式の違いによる差はそれほどない。

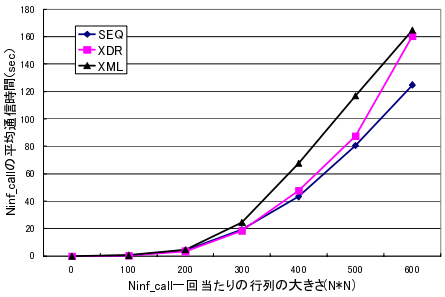

図 8 global 環境における呼び出し間隔一定のモニタリング

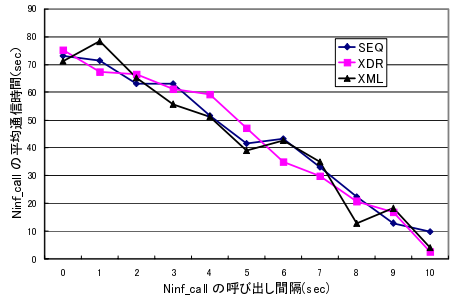

図 9 local 環境におけるデータサイズ一定のモニタリング

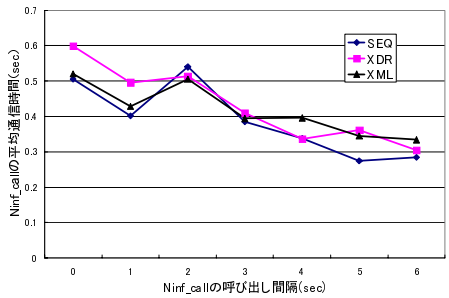

図 10 global 環境におけるデータサイズ一定のモニタリング

5.4 考　　察

Micro Benchmark の結果から、XML のパーズコストは代替として考えられる他の手法と比較して明らかに大きいことがわかる。しかし、実際にモニタリングシステムが運用される状況を模した実験では、表現手法が性能に与える影響は認められるものの、十分に小さく許容できる範囲内であることがわかった。

また、Grid RPC 向けの event を定義した経験から、event の表現手法としての XML スキームは十分に強力で、拡張性に富むことがわかった。

6. 終 わ り に

Global Grid Forum 内の組織の一つである Grid Performance Working Group が提案している、モニタリングシステムの基本的なアーキテクチャと XML によるデータ形式に関して 1) アーキテクチャのスケーラビリティ、2)XML を用いたデータ表現のコスト、3) データ形式の拡張性、について検証するために、GridRPC システムである Ninf 上に提案アーキテクチャの一部を実装し、評価をおこなった。その結果、アーキテクチャが現実的な設定範囲内では十分スケーラブルであり、データ形式の拡張性も十分であることを確認できた。

今後の課題として、まず一つめに提案されたアーキテクチャのうち未実装である機能を実現する。即ち、consumer に event に関する情報検索を提供する Directory Service や モニタリングデータを実際に計測する sensor 等を実装することが挙げられる。また、2 つめに今回の実験では広域な環境として WAN で接続された 2 つのサイト間で検証を行ったが、複数のサイトを接続した更なる広域分散環境で検証を行うことが挙げられる。

参 考 文 献

1) GridForum Grid Performance WorkingGroup. http://www-didc.lbl.gov/GridPerf/.
2) Ninf. Network infrastructure for global computing. http://ninf.etl.go.jp/.
3) Network Weather Service(NWS). http://nws.npaci.edu/NWS/.
4) Active Measurement Program(AMP). http://amp.nalanr.net.
5) Surveyor. http://www.advanced.org/surveyor.
6) GridGlobal Forum. http://www.gridforum.org.
7) Grid PerformanceWorking Group. http://www-didc.lbl.gov/GridPerf/.
8) GridForum Grid Performance WorkingGroup. Schemas for Grid Performance Events, October 2000. http://www-didc.lbl.gov/GridPerf/paper/EventSchemas.pdf.
9) GridForum Grid Performance WorkingGroup. White Paper: A Grid Monitoring Service Architecture(DRAFT), 2000. http://www-didc.lbl.gov/GridPerf/paper/GMA.pdf.
10) 関口智嗣, 佐藤三久, 長島雲兵. ネットワーク数値情報ライブラリ: - ninf の設計 -. 情報処理学会研究報告 HPC, Vol. 94, No. 94-HPC-52, p. 127, 1994.
11) 中田秀基, 佐藤三久, 関口智嗣. ネットワーク数値情報ライブラリ ninf のための rpc システムの概要. TR 95-28, 電子技術総合研究所, 1995.