2K-7

Ships1 におけるシステム統合リポジトリの設計

櫻井 一欽† 加藤 悠† 大谷 真†
湘南工科大学 情報工学科†

1. はじめに

Ships1(Shonan Institute of Technology Parallel System 1)は、低価格で中小規模応用を目指す並列マシンである。Ships1 では、一般の高度科学技術計算向け並列マシンとは異なり、各ノードの構成が均一ではないとの特徴がある。したがって、ジョブ振り分けなどの並列制御においてはノード構成を示す静的情報と CPU 負荷などの動的情報を総合的に処理する必要がある。本論文では、Ships1 情報の統合管理を行うシステム統合リポジトリについて、その考え方、モデル設計、モデル実装、API 設計及びプロトタイプ実装結果を述べる。

2. 情報管理

Ships1 の制御や管理には、ハードウェア/ソフトウェアの構成情報や実行時のノードやネットワークの稼動情報を集める必要がある。このような情報は、長時間変化しない静的情報と時々刻々と変化する動的情報の二種類に分けられる。

- 静的情報：各ノードのハードウェア構成情報 ネットワーク構成情報、ネットワークアドレス情報など
- 動的情報：各ノードの CPU 使用率、ハードディスク利用率、メモリ使用率など

これらの情報を使用し、システム監視、負荷バランス、動的ジョブ割付、システム制御などが動作する。一般にこれらの情報は別々に管理されることが多い。Ships1 では、システムデザインの統一化を目的に、これらのシステム情報を単一のリポジトリで管理することした。(図 1)

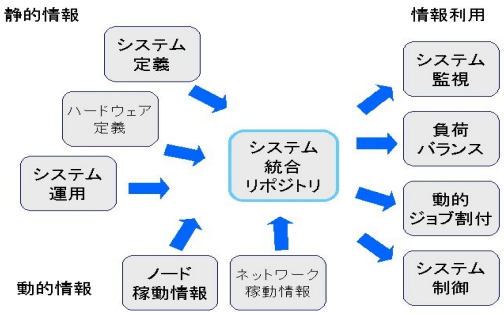

図 1:システム情報とリポジトリ

3. システム統合リポジトリ

図 1 のようなシステム統合リポジトリを設計するには、まず、情報のモデル化が必要である。

Design of integrated system repository for Ships1

Kazuyoshi Sakurai, Yu Katou, and Makoto Oya
† Department of Information Science, Shonan Institute of Technology

モデルでは、大きく分けて静的情報を示す Physical Data モデル、動的情報を示す Dynamic State モデルで構成される。

モデルにおいては、Node の識別方式が重要である。ノード構成情報には、静的情報内で変化しない情報が好ましい。絶対に変化しない絶対名を採用した。絶対名はノード固有に存在する固有名で、ノードを完全に識別するものとして、用意したものである。図 2 は、システム統合リポジトリを簡略に示したものである。

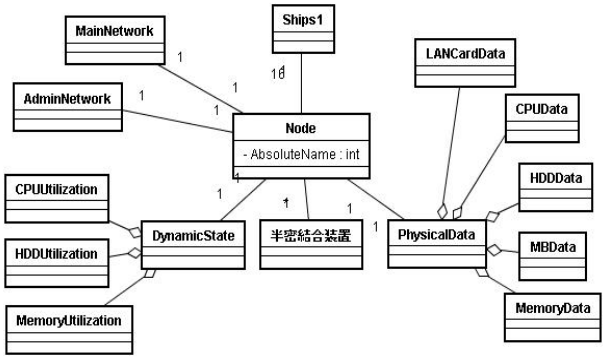

図 2:システム統合リポジトリモデル(簡略化版)

Physical Data には、ハードウェア構成情報の M/B 情報、HDD 情報、M/B 情報、CPU 情報、LAN 情報、Memory 情報、ネットワーク情報のホスト名、MAC アドレス、IP アドレスなどがある。

Dynamic State は、Node 負荷情報の CPU 使用率、HDD 使用率、Memory 使用率である。また、今回、新規開発される半密結合装置については、ネットワークインターフェイスと同様に扱えるために LAN デバイスと扱ってもよいが、すべてのノードに無いために、扱いは別ものとした。

4. モデルの実装

図 2 で示したモデルを実装モデルにする。DB には MySQL を採用した。MySQL は、オープンソフトウェアで、View やストアドプロシージャなどにも対応しているので採用した。表 1 は設計した主要なテーブルの一覧である。

表 1:主要なテーブル一覧

テーブル名	説明
cpuspec	CPU情報
hddlist	接続ディスクのリスト
hddspec	HDD情報
lanspec	LAN情報
maker	メーカ情報
mbspec	Mb情報
memoryspec	Memory情報
node	Node情報
nodespec	Nodeスペック情報

動的情報には、SNMP(Simple Network management System)で取得できる値を採用した。

静的情報のディスク情報は、今後、各ノードに複数台導入される可能性ことを考慮し、別途にテーブルを用意し各ノードのハードディスクリストを管理することにした。

動的情報は詳細なシステム稼動情報が多量になることを考慮して、情報の保存方式を設計した。短期的な情報に関しては密に保存し、長期保存されている情報に関しては疎に保存する方式を取り、DB の情報を定期的に調査して保存間隔を調整する。

5. API

Db と O/R マッピングしたうえで、モデルに従ってリポジトリにアクセスするオブジェクト指向型の API を考案した。以下で API の使用方法を説明する。はじめに主体となる Ships1 クラスのインスタンス化が必要である。Ships1 クラスには、Node 固定に必要な Node 情報取得関数が存在する。次に、各 Node のインスタンス化を行うが、ここではインスタンス化時にどの Node を指し示すかは決定しない。Node クラス認識方式は、Node クラスをインスタンス化した後に、bind_by_関数を使用して行う。bind_by 関数には、Node 固定化を行う値として Node 名、MAC アドレス、IP アドレスを採用した。図 3 に Node 固定化の API 使用例を示す。

```
$top = new Ships1( ); // 初期動作
$list = $top->get_Node_Name( ); //Node 名一覧取得
$top->Node1 = new Node( ); //Node の生成
$top->Node1->bind_by_NodeName($list[1]); //  固定化
```

図 3:Node 固定化方式の例

Node 固定化後、静的・動的情報の取得が可能となる。静的情報は Physical Data のインスタンス化時に同時にインスタンス化された MB Data, CPU Data, HDD Data, Memory Data, LAN Data 内関数を使用することにより、各 Node の情報を取得することが可能である。Physical Data の固定化方式は、Physical Data クラスのインスタンス化時に Node のインスタンスを値として渡す。図 4 は静的情報取得の API 使用例である。

```
$Physical = new PhysicalData($top->Node1 ); // 初期動作
/*Node の MB の製品名とチップセット名の取得 */
$MB_Name = $Physical->MB->get_ProductName( );
$MB_CHI = $Physical->MB->get_Chipset();
/*Node の HDD 情報取得 */
$HDD1_Name = $Physical->HDD[0]->get_ProductName( );
$HDD1_Size = $Physical->HDD[0]->get_Size();
$HDD2_Name = $Physical->HDD[1]->get_ProductName( );
$HDD2_Maker = $physical->HDD[1]->get_Maker();
```

図 4:静的情報の取得例

Dynamic State クラスもインスタンス化時に同時にインスタンス化された CPUUtilization, HDDUtilization, MemoryUtilization 内関数を使用することで各 Node の動的情報を取得することが可能である。情報は、即時的な性能が要求されないものについては、DB から取得し、性能が要求されるもについて、メモリ上から取得する方式とした。DB からは情報保存間隔を考慮し、近似値を用いて行う。情報取得関数は 3 通り用意し、

・今現在の値を取得する方式
・日付を指定してその日の情報を取得
・取得範囲を指定して情報取得を行う方式

3 つ目の方式は、過去の情報になるに連れ、間隔が広くなっているので、情報を近似値で取得するようにした。図 5 は動的情報の取得の API 例である。

```
$Dynamic = new DynamicState($top->Node1 ); // 初期動作
/* 現在の使用率 */
$CPU_t = $Dynamicl->CPU->get_cpu( );
/* 指定日の情報を取得 */
$CPU_d = $Dynamic->CPU->get_day_cpu("070101");
/* 指定範囲の情報を取得 */
$Dynamic->CPU->set_cpu("070101", "070102");
$CPU_r = $Dynamic->CPU->get_next_cpu( );
$CPU_r = $Dynamic->CPU->get_next_cpu( );
```

図 5:動的情報の取得例

6. まとめ

PHP、MySQL を使って API を実装し、実際に動作することを確認した。これにより、モデルの有効性、O/R マッピングの妥当性が検証できた。

静的情報と動的情報を単にアクセスすることは一般に行われているが、同一モデル、同一 API で利用者側からみたとき区別なく扱うことができるようになった。

API を実装することにより、情報取得の方式は完成した。今後、高速並列計算に向けて、高速に動作する API の実装などが必要である。

参考文献

[1]大谷真, 松原裕人, 櫻井一欽, 加藤悠, 中小規模並列コンピュータ Ships1 の開発, 湘南工科大学紀要, Vol.41, No.1, 2007

[2]宮川伸也, 佐治伸之, 工藤裕, 田崎英明, ビジネスグリッド技術解説, 情報処理学会, Vol.45, No.9, pp.953-961, 2006

[3]Mark A. Miller, SNMP インターネットワーク管理, 金子卓也訳, 木部真也訳, 翔泳社, 1998