B-034

グリッド環境における異種スケジューラの連携の提案

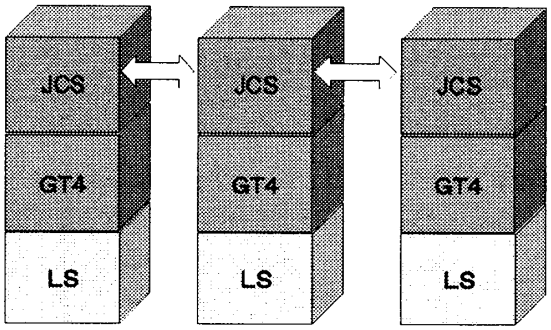
JSC：Job Scheduling Coordinator of heterogeneous local schedulers in the Grid environment

高橋直人†　　内田啓一郎†
Naoto Takahashi　　Keiichiro Uchida

1．はじめに

近年、ネットワークに接続されたコンピュータが増え、それらを有効活用するグリッド技術の研究が盛んである。しかし、ネットワークに繋がれていても組織によって異なるローカルスケジューラを用いて運用している場合が多い。そのためユーザはシステムごとに操作性の違いやジョブの記述などを理解し、習得する必要がある。そこで本研究は、異機種分散となるグリッド環境において、システムの違いを意識することなくローカルスケジューラを統合する上位スケジューラ、ジョブスケジューリングコーディネータ(以下 JSC)を提案する。JSC は、システムに大幅な変更を加えることなく、既存の技術を組み合わせることによりオープンなアーキテクチャを実現する。

JSC とローカルスケジューラを繋ぐインターフェースとして Globus Project[1]が提供する GlobusTookit4 (以下 GT4)を利用する。また本提案ではローカルスケジューラとして OpenPBS[3], SunGridEngine[4], Condor[5]で運用されているクラスタシステムでの利用を前提とする。各システムの位置付けを図１に示す。



LS:ローカルスケジューラ

図１：位置づけ

2．JSC の概要

JSC はユーザにスケジューラによって問題となるシステムの操作性の違いを埋めるためのシステムである。JSC の目的とする操作性の統一である。ジョブ記述には現在 GGF(Global Grid Forum)[2] で標準化作業が進められている JSDL(Job Submission Description Language)を採用する。

本提案で利用する GT4 にはスケジューリング機能は無い。そのため JSC で全体のスケジューリングを行い、GT4 は JSC とローカルスケジューラのインターフェースとして振舞う。JSC はセンターサーバを持たず各ホストに配置される。ユーザは任意の JSC にジョブを投入する。

3．ユーザ認証及び通信レイヤ

ユーザやホストの認証には GT4 が提供する機能を利用する。また JSC 同士の通信には、GT4 の提供する通信レイヤと認証システムを利用する。GT４の通信レイヤは Web サービスが用いられており、Firewall の問題を解決できる。

4．実行ユニットアーキテクチャ

JSC の実行ユニットアーキテクチャを図２に示す。

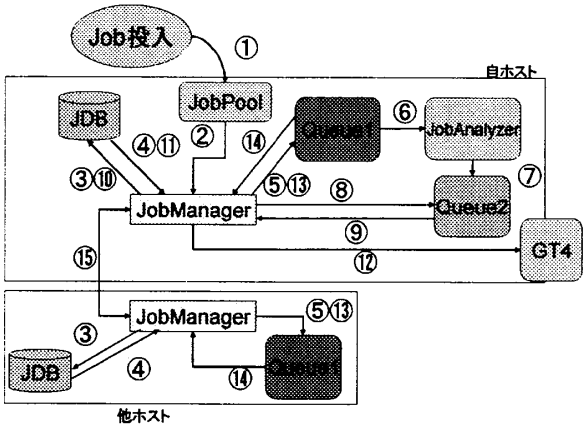


図２：実行ユニット

JSC は、次のコンポーネントと外部アプリケーションとの接続のためのインターフェースから成る。

4．1　JobManager

JobManager はスケジューリング機能をメインとし、各コンポーネントを繋ぎジョブを処理する中核となる機能である。

JobManager は同じポリシーの元で、利用可能なサイトを記したホストリストを持つ。JobManager はホストの状態を外部に公開することにより他ホストから状態問い合わせに対応する。また他ホストへの問い合わせも行い、自ホストよりも負荷が低いホストがあればジョブの転送の可能性を判断する。負荷の判別は Queue１に入っている待機ジョブの数により決定する。また投入時間による優先順位を付け何時まで経っても実行されない状態を回避する。

4．2　JobAnalyzer

JSDL からローカルスケジューラ向け GT4 用のジョブファイルを生成する。抽象的なジョブ記述から具体的な実行用ファイ

†神奈川大学大学院理学研究科

ルを生成することにより異なる環境でも同じジョブを実行できるようになる。

４．３　JobPool

投入されたジョブ記述ファイルと必要なデータファイルにIDを付け区別を行う。

４．４　Queue１

ユーザによって投入されたジョブが最初に入れられるQueue。ここへ投入されたジョブはJobAnalyzerに送られるか、通信モジュールを介して他ホストへ送られる。投入できるQueueの数はシステム性能に応じて設定する。

４．５　Queue２

JobAnalyzerによって生成されたファイルのQueue。実行待ちジョブが入れられ、GT4に送られる。Queueの数はQeueu1と同様に設定する。

４．６　JDB

JDB（Job Database）はジョブの実行に必要なファイルはすべてデータベースに格納する。ジョブファイルの他にホスト情報などのデータも格納する。このデータデースは定期的に他ホストのデータベースと同期を取り、後述する障害対策に使われる。

５．情報収集ユニット

情報収集ユニットのアーキテクチャを図３に示す。

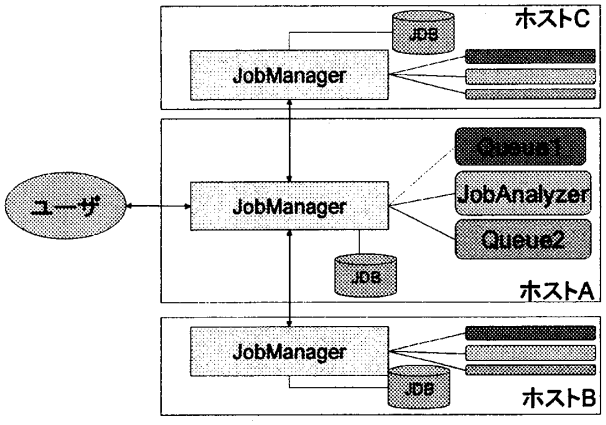


図３：情報収集ユニット

JobManagerによって定期的に実行状態や負荷情報、他ホストの状態などが収集されDBへ書き込まれる。ユーザはDBからホスト情報や利用できるアプリケーション等を読み出し、それらの情報からジョブを記述する。

６．JSCの実行フロー

JSCの実行データフローを図２に対応させて説明する。

① ジョブを投入する。
② ID付けされたジョブを読み出す。
③ ジョブデータをJDBに書き込む。
④ JDBからジョブ記述ファイルを読み出す。
⑤ ジョブ記述ファイルをQueue１に入れる。
⑥ Queue１のジョブ記述ファイルをJobAnalyzerに入れる。
⑦ JobAnalyzerによって処理されたジョブ記述ファイルをQueue2に入れる。
⑧ データの読み出し要求。
⑨ ジョブ記述ファイルを読み出す。
⑩ ⑨で読み出したファイルに対応する実行用データを要求。
⑪ ⑨に対応するデータの読み出す
⑫ 実行ファイル類をGT４に投入する。
⑬ Queue1の状態を問い合わせる。
⑭ Queue1の状態を応答する。
⑮ 他ホストの状態を問い合わせ、負荷が自ホストより低ければジョブを転送する。

７．障害対策

ジョブファイルは、ジョブ投入時にDBへ書き込まれると共にホストリストにあるホストから任意の他ホストへコピーが転送される。２ホスト以上がジョブファイルを持っていることによりファイルの消失を防ぐことが可能になる。

またタスクの再実行機能としてホストリスト内にあるホストの反応が無くなった場合に、JobManagerがダウンしているホストのジョブの再振り分けを行う。これによりシステムダウンによりタスクの消失や停止を回避することを可能とする。システムダウンより復旧したホストは、他のサイトから実行情報を取り出し、自ホストでの再実行が不要であれば実行中であったジョブを破棄する。また他ホストで未実行であれば自ホストで実行し、他ホストでの実行命令に破棄を通知する。

８．まとめ

本提案では既存システムとGlobusToolkit4にJSCを加えることで異なるスケジューラ同士でも差異なく利用できる手法を提案した。本提案では他ホストを跨ぐMPIの実装や、連続タスクへの対応も必要性が検討課題である。ユーザインタフェース等のユーザ側のシステムも検討段階であり、ワークフローなど可視的な部分への対応も今後の課題である。

参考文献

[1] Globus Project : www.globus.org
[2] Global Grid Forum : www.gridforum.org
[3] OpenPBS : http://www.openpbs.org
[4] SunGridEngine : http://gridengine.sunsource.net
[5] Condor® Project : http://www.cs.wisc.edu/condor/
[6] NAREGI : http://www.naregi.org