2000 年 9 月 8 日

サイエンティフィック・システム研究会
科学技術計算分科会　第一回会合資料

VPP800 の新しい運用形態を求めて

平野彰雄*　浅岡香枝*　金澤正憲*
hirano@kudpc.kyoto-u.ac.jp　asaoka@kudpc.kyoto-u.ac.jp　bwv147@mbox.kudpc.kyoto-u.ac.jp

*京都大学大型計算機センター

1. はじめに

京都大学大型計算機センター（以下、本センターと呼ぶ）は、全国の大学、高等専門学校などの研究者が学術研究に伴う科学技術計算および情報処理を行うための全国共同利用施設である。大規模科学技術計算需要に応えるため、1999 年 3 月にスーパーコンピュータ Fujitsu VPP500/15 (以下、VPP500 と呼ぶ）から Fujitsu VPP800/63（以下、VPP800 と呼ぶ）に更新してサービスを行っている。

本センターの計算サービスは、運用時間帯であれば研究者が自由にジョブを投入できるオープン利用方式をとっており、様々な研究分野の研究者によりプログラム開発、プロダクション・ランなどのプログラムが混在して投入される。このような状況では、ジョブのターンアラウンドを保証し、かつ、システムの運転効率を高めるためには、ジョブのスケジューリングが重要になってくる。

本稿では、まず、本センターの VPP800 の構成を紹介し、つぎに、これまでの運用形態の変遷と課題を整理して、最後に、現在、検討している自動運転スケジューラの機能と実現方式について述べる。

2. VPP800 の構成

VPP800 のシステム構成を図 1 に示す。

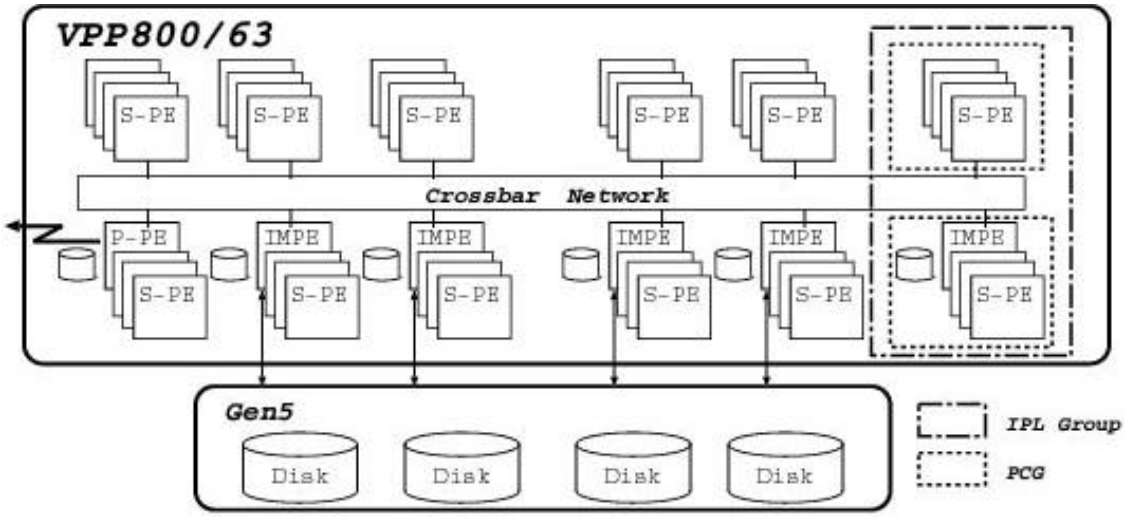

図 1：VPP800 のシステム構成

VPP800 は、63 台のプロセッサ（PE: Processing Element）から構成され、PE 間をクロスバーネットワークで結合した並列計算機である。各PE は、最大ベクトル演算性能8GFLOPS、メモリ 8GB をもったベクトルプロセッサであり、システム全体では総ベクトル演算性能 504GFLOSP、総メモリ容量 504GB である。

PE には、入出力機能を備えた IO-PE と入出力機能を持たない S-PE（Secondary PE）がある。63 台の PE のうち 8 台が IO-PE で、残りの 55 台が S-PE である。

システムの起動は、IPL-GROUP という単位に行われ、１つの IPL-GROUP は、1 台の IO-PE と 7 台の S-PE で構成される。また、IPL-GROUP の IO-PE を IMPE と呼び、8 台ある IMPE のうち、1 台を P-PE(Primary PE)と呼びシステム全体を統括する機能を持つ。

IPL-GROUP は、PCG（Power Control Group）と呼ばれる単位に分かれており、このPCG 単位に電源供給の制御が可能となっている。VPP800では、4台の PEで１つの PCG を構成している。したがって、IO-PE を含まない 8 つの PCG すべてを停止し、32PE 構成のシステムまで縮退することも可能である。

3. 運用形態の変遷と課題

3.1. 導入と正式サービス開始

VPP800 運用は、1999 年 3 月から 5 月末までは、ベクトル計算機 VX からのジョブ投入という形態で NQS(Network Queuing System)バッチ専用のシステムとして運用していた。1999 年 6 月から VPP800 の P-PE に直接ログインしてTSS 会話型での利用およびジョブ投入という形態に移行し、ここから VPP800 の正式サービスとしている。

正式サービス開始時の NQS のキュー名と許可量を表１に示す。

表１： キュー名と許可量(1999 年 6 月～2000 年 3 月)

キュー名	最大 PE 数	CPU 時間	メモリサイズ	備　　考
cl	1	30 分	2GB	コンパイル専用
d	1	1 時間	7GB	1PE ジョブ
e	1	6 時間	7GB	
f	10	1 時間	7GB×10	10PE ジョブ
g	10	6 時間	7GB×10	
h	40	6 時間	7GB×40	40PE ジョブ

この時点での NQS キューの定義は、それまでの運用の継承を最優先に考えて設計した。ここでは、コンパイルジョブ専用のキューcl と 40PE まで使用可能なキューｈを新たに設けた。

ジョブのスケジューリングとしては、NQS-JM（Job Manager）の機能を用いて、つぎのような設定を行った。

1) 同じ利用者ジョブでシステムが占有されないために、NQS キューでの利用者リクエストの同時実行多重度を 1 に設定した。
2) MRFS（Memory Resident File System）をサービスするために標準値を０GB、最大値を 7GB に設定した。

3.2. システムのレベルアップとＮＱＳ-ＪＳの適用

1999 年 10 月に OS の PTF アップが提供された。これに伴い新規機能追加に合わせて、つぎのような運用形態に移行した。

1) ファイルシステムの移行

これまで大容量ファイルシステムとしては、VFL しか使えなかったために 1 利用者当たり 1,500 個という大きな制約があったが、これを新たに提供された FPFS（Flexible and high Performance File System）に移行し、1 利用者当たりのファイル許可量を 10 倍の 15,000 個まで拡張した。

2) NQS-JS（Job Scheduler）の適用

これまでのジョブのスケジューリングは、基本的に NQS のキューおよびコンプレックスキューの Run_Limit で制御し、資源待ちが発生しないような制御を行っていたが、NQS-JS 機能が提供されたので、これに置き換えた。NQS-JS はシステムの空き資源がない限りジョブをスケジューリングしない。

なお、NQS-JM、NQS-JS を組み合わせれば、より細かいレベルでのジョブスケジューリングも可能であるが、効果的にやるには、キューの許可量の変更など利用者に直接影響する部分での運用の変更も必要となる。しかし、このような運用変更を年度の途中に実施することは、利用者が混乱するだけなので見合わせた。

3.3. 運用形態の変更と新たな課題

2000 年 4 月からの新年度運用に合わせて、それまでの運用を見直して、運用形態の大幅な変更を行った。具体的には、つぎのものである。

1) jobexec コマンドによる会話型並列サービス
2) NQS の許可量の見直し
3) SHARE モードでの運用

3.3.1. 会話型並列実行のサービス

これまで並列ジョブは、NQS 経由でないと認めていなかった、しかし、Try & Error での並列プログラムの開発作業は、会話型のほうが効率よい。したがって、jobexec コマンドに対する１）システム内での同時実行コマンド数制限、２）利用者当たりの同時実行コマンド数制限、３）jobexec コマンドの経過時間打ち切り機能などを要望していたが、これらの機能が 3 月末に提供されたので、4 月から jobexec コマンドによる会話型並列サービスを開始した。

本センターでの jobexec コマンドの許可量および制限は、つぎのように設定している。

1) jobexec コマンドの許可量
   - 最大 CPU 時間 30 分、最大 PE 数　４、経過時間　1 時間
2) 制限値
   - システム内多重度　４
   - 利用者当たりの実行コマンド数　1

3.3.2. ＮＱＳ の許可量の見直しと運用形態の変更

4 月からの NQS の許可量を表２に示す。

表２：キュー名と許可量（2000 年 4 月～）

キュー名	最大 PE 数	CPU 時間		メモリサイズ		備　　考
		標準	最大	標準	最大	
cl	1	－	30 分	－	2GB	コンパイル専用
d	1	－	1 時間	3GB	7GB	1PE ジョブ
e	1	6 時間	20 時間	3GB	7GB	
f	10	－	1 時間	3GB×10	7GB×10	10PE ジョブ
g	10	6 時間	20 時間	3GB×10	7GB×10	
h	40	6 時間	20 時間	3GB×40	7GB×40	40PE ジョブ

基本的な変更点は、メモリサイズを標準3GB、最大 7GB とし、また、CPU を最大 20 時間まで緩和したことである。

メモリサイズの 3GB は、それまでのアカウントデータを分析して、8 割以上のジョブが 3GB 以下で実行されている事実からこの値を採用した。これらの標準値、最大値の管理は、NQS-ＪＭ の資源割り当て管理機能を用いている。

また、それまでの運用では、最大値だけで管理しており、利用者に対してジョブ投入時に必要なジョブ資源を明示することを要求してこなかったが、メモリ量、CPU 時間、および PE 数を qsub コマンドで指定するように指導するようにした。

3.3.3. ＳＨＡＲＥ モードでの運用とジョブスケジューリング

4 月からの運用では、NQS キューの許可量を見直すとともに並列、非並列ともにキューの定義を SHARE モードに変更してサービスを開始した。なお。並列ジョブのバリアモードは、高速バリアモード（バリアモード０）である。

基本的なジョブスケジューリング戦略は、つぎのようなものである。

1)　ジョブクラスリスト

- 非並列キューは、基本的に P-PE を除く 7 台の IO-PE にだけに割付ける。
- 並列キューは、基本的に 55 台の S-PE に割り付ける。

2)　NQS-ＪＳ の設定

- max non-parallel=2 を指定し非並列ジョブが最大２ジョブ同時に実行されるようにし、また、空き PE がある場合にはジョブが重ならないように pack=no を指定した。
- また、CPU 時間、メモリ量、PE 数のファクタで要求資源の少ないジョブが優先的に実行され、かつ、実行待ち時間により多くの資源を必要とするジョブが不当に永く待ちになることを防いでいる。

3)　Run_Ｌｉｍｉｔによるスケジューリング

- 40PE を使うキューｈのジョブは、他の並列ジョブ（キューf,g）が複数ある限り、必要な PE 台数を確保できないので、実行待ち時間を調べ Run_Limit を調整することで長時間待ちを回避している。

3.3.4. SHARE モードの問題と対策

ジョブが混み始めた5月中旬、利用者から同じジョブでCPUが非常にばらつくという申告を受けた。

調査すると二つの並列ジョブの割り付けPEが部分的に重なった場合の問題であることが判明したので、並列ジョブキューの実行モードをSIMPLEXに変更して追試を行った。

追試の結果、本センターのようにジョブ課金を行っている場合、バリア同期の方法をCPU_LOOPからSYS_POLLにしないとSHAREモードでの運用は困難であることが明らかとなり、SYS_POLLに切り替える場合の問題点について検討した。明らかとなった問題点は、つぎのようなものである。

1) CPU_LOOPとSYS_POLLの切り替えは、プログラム実行時の環境変数での指定であり、センターとして一括して変更できるような枠組みがいる。
2) バリア同期の方法が環境変数で指定可能になった時期がFortran/VPP,HPF、メッセージパッシングライブラリ（MPI、PVM）で異なる。
3) SYS_POLLに切り替えた場合、システムコールのオーバヘッドにより性能劣化が起こるので、ベンチマーク的なジョブにはまずい。
4) PA(Performance Analyzer)でデータ転送に関するデータを測定する場合、SIMPLEXモードの実行が必須。

また、これらの問題に対する対処は、つぎのようなものである。

1）の問題に関しては、環境設定ファイルにFLIB_SYSPOLL（Fortran/VPP,HPF）およびVPP_POLL（MPI,PVM）を指定し、並列ジョブのバリア同期の方法をデフォルトでSYS_POLLに切り替える。2）の問題については、センターで導入したアプリケーションパッケージに関する調査では問題がなかったが、センター外で作成されたモジュールを利用者が実行する場合、注意が必要である。また、3）、4）の問題に関しては、本来、ジョブの投入時にqsubコマンドのオペランドで実行モードが選択できればよいが、NQSではこのような機能はない。したがって、メモリサイズで7GBを指定して貰いPEを占有して実行させるとともに利用者が環境変数でバリア同期の方法をCPU_LOOPに切り替えて実行するようなアナウンスが必要である。

SHAREモードでの運用は、キューhのような高並列ジョブのスループットの改善には必須であると考えている。ここで上げた問題点および対処方法を整理して、利用者に対するアナウンスの後、適当な時期からSHAREモードでの運用に戻すことを考えている。

4. 自動運転スケジューラ

4.1. 基本概念と機能

本センターでは、これまでシステム運転サービスのためにスケジューラを独自開発する一方でNQS-JM,NQS-JSを使って、利用者ジョブの管理、ジョブのスケジューリングを行ってきた。

しかし、このようなシステムでは、つぎのような問題がある。

1) 定期保守など計画的なシステム停止の場合、指定時間までに確実にジョブ処理を終えるような枠組みが無く、人手に頼っており、最悪、強制停止、ジョブのリスタートが発生している。
2) また、システムが混んでくると40PEを必要とする高並列ジョブ（キューh）のターンアラウンドの保障が困難である。これも人手でJob_Class_List、Run_Limitを調整している。

したがって、これらの問題に対処するために、自動運転スケジューラの実現を検討している。
この自動運転スケジューラの基本的な機能としては、つぎのものである。

1) 運転スケジュール管理機能
システムサービスのスケジュールを管理して、システム定期保守（毎週火曜日17:00）や空調設備の定期点検（隔月第3日曜8:30）時には、全てのジョブ処理を停止して、システムを停止する。

2) ジョブリクエストの管理機能
投入されたジョブが要求しているシステム資源を把握して、システム停止時間までに終了しないと予測されるジョブはスケジュールされないようにホールドする。
また、状況を判断してジョブをリリースする。

3) システムの動的構成変更機能
システム運転スケジュールおよび実行待ち、実行中ジョブの状況をもとに、Job_Class_List、Run_Limit,NQS-JSのパラメータなど動的に変更する。

4.2. スケジューリングの戦略と実現方式

4.2.1. スケジューリングの戦略

自動運転スケジューラの実現方式を説明する前に、本システムの基本となるスケジューリングの戦略について紹介する。

すなわち、ジョブのターンアラウンドの保障とシステムの運転効率の向上といった相反する要求を満足するスケジューリングを実現することは、ある意味困難である。しかし、この間のシステムの利用状況を整理すると、つぎのような特徴がある。

- 1日を時間帯で見れば15:00以降、夜中2:00までに利用が多く、他の時間は比較的すく傾向にある。
- 週単位に見ると月曜は比較的すいており、木曜以降週末の利用が多くなる。
- また、これは本センターの負担金制度ともからむ話しであるが、定額利用制度期間の最終月には利用が集中してしまう。

したがって、このような利用状況を背景に、つぎのような戦略を妥当なものとする。

1) システムの運転カレンダーをジョブのスケジューリングで最優先のパラメータとする。
2) 混雑時は、ジョブが要求するCPU時間ぐらいの実行待ちは妥当であると判断し、要求するCPU時間が短いものを優先してスケジューリングする。
3) キューhのような多くの40PEを必要とするジョブは、極力利用の少ない時間帯（例えば、深夜2時以降、週末から日曜）に実行されるよにうスケジューリングする。

4.2.2. 実現方式

自動運転スケジューラシステム（以下、本システムと略す）の実現方式としては、NQSとのより密な連携のためにNQS出口ルーチンの利用も検討したが新規にプロセスを起動できない、SIGNALマスクがさわれないといった大きな制約があること判明したので、独立したデーモンプロセスとして作成することにした。

実現方式の概要は、つぎのようなものである。

1) NQSは起動時からGlobal_Batch_Limitをゼロに設定することで、全てのジョブスケジューリングを本

システムが制御できるようにする。

2) 本システムのスケジューリング処理には、ジョブの受付け、終了、キャンセルなどジョブの状況変化を契機にする処理とタイマーにより一定で行う二種類がある。前者は、NQS のジョブ処理と本システムを密に連携させるために追加した機能で、NQS の出口ルーチンを改造して、新たなログメッセージを NQS ログに出力させ、これを本システムが注視する形で実現している。
3) ジョブが新たに投入されると、そのジョブの要求している CPU 時間を調べ、システム停止時刻までに終了しないと判断したジョブは qhold コマンドにより、オペレータホールド属性に遷移させる。また、一旦、システム停止時刻までに実行可能と判断され、実行待ちとなっているジョブの CPU 時間を調べ、終了しないものは同様にオペレータホールド属性に遷移させる。さらに、Global_Batch_Limit を一定時間大きくすることで、実行待ちジョブを実行させる。
4) また、ジョブの実行終了や実行中ジョブがキャンセルされた場合も、実行待ちジョブの状況を調べて、Global_Batch_Limit を操作することでジョブを実行させる。
5) タイマーにより起動された契機では、まず、実行待ちジョブの待ち時間と要求資源を調べて長時間待ちになっているジョブがあれば、実行中ジョブの残りの実行時間を算出して、残り時間の少ないジョブが使っている PE を調べて、Run_Limit と Job_Class_List を変更し,長時間待ちジョブが実行できるように構成変更を行う。

プログラムの開発は、基本的に perl を用い、ジョブ状況の確認には、システムで提供されるコマンドを利用して実現することを考えている。

5. おわりに

VPP800 の導入から現在までの運用形態の変更と新たに発生した問題とその対策について紹介した。また、現在、検討している自動運転スケジューラの機能と実現方式について説明した。

本センターのようにオープン利用を採っている場合には、利用者に対する運転カレンダーの衆知が必要であり、さらに、ここで紹介した自動運転スケジューラシステムがとるスケジューリング戦略も利用者に広報して、ジョブが必要する CPU 時間や PE 数などをより正確に利用者に指定して貰う方向に誘導しないと、十分に効果を発揮できないのは明らかでる。

また、最近、大規模な並列シミュレーションを行われている利用者から 1 週間単位の CPU 時間のジョブを実行させたいという要望がよせられている。メモリが大きくなり大規模なモデルが演算できるが、現状の最長 20 時間という CPU 制限では、プログラムの中断、再開の入出力コストが多いという主張である。

センターとしては、このような長大ジョブ認め、かつ、使いやすいスパコンサービスを実現するために、現状のシステム運転サービス時間や定期保守の体制も含めて見直すことも必要であると考えている。

[参考文献]

[1] 平野彰雄,植木徹,赤坂浩一,浅岡香枝,金澤正憲:京都大学における VPP800 の運用と性能評価,SS 研科学技術計算分科会，1999 年 8 月

[2] 植木徹,平野彰雄,金澤正憲:VPP800 における運用効率とジョブスケジューリング、京都大学大型計算機センター研究発表報告集 No.15, 2000 年 3 月

[3] 富士通株式会社:UXP/V ネットワークキューイングシステム説明書 V20 用,1999 年 10 月

[4] 富士通株式会社：UXP/V ネットワークキューイングシステム-JS 説明書 V20 用,1999 年 10 月

[5] 富士通株式会社：UXP/V ネットワークキューイングシステム-JM 説明書 V20 用,1999 年 2 月

[6] 富士通株式会社:UXP/V 並列ジョブスケジューリングガイド(UXP/V V20 対応版),2000 年 6 月