# 複数ＯＳからのスーパーコンピュータの利用方式の開発

平野彰雄、金澤正憲
(京都大学大型計算機センター)

概　要

京都大学大型計算機センターでは，1995年1月に新ベクトル並列スーパコンピュータVPP500を導入した。このVPP500のサービスにあたって，汎用OSのホストシステムおよびUNIXシステムの両方からジョブ依頼するためのシステムを開発した。本論文では，ホストシステムからのジョブ投入のためにM-VPP連携機能およびUNIXシステムからのジョブ投入のための利用者管理機能とジョブ実行制御機能について述べ，その実現方式について報告する。

# Design and implementation of job control method among two different FEPs and a supercomputer

Akio HIRANO, Masanori KANAZAWA
(Data Processing Center, Kyoto University)

Abstract

The new vector parallel supercomputer VPP500 was introduced in our Center last year. We have designed and implemented the system in which both the IBM-like computer system and the UNIX system can submit batch jobs to the supercomputer and control job execution under respectively Job Entry Subsystem and Network Queuing System.

## 1　はじめに

京都大学大型計算機センターは，計算機利用のための全国共同利用施設であり，登録利用者は毎年5,000名を超え，高速大規模な科学技術計算のための計算機需要も年々増加する傾向にある。この高速大規模な演算需要に応え，これらのジョブのスループットの大幅な改善を目的に，1995年1月にスーパコンピュータのリプレース行った。

本センターでのスーパコンピュータの導入は1984年のベクトル計算機VP100が最初であり，ベクトル計算機のオペレーティングシステム(OS)は，IBM-MVS系の汎用OS (MSP)であった。今回のリプレースでは，それまでベクトル計算機VP2600一台であったものをベクトル計算機VP2600/10Eと分散メモリ型並列計算機VPP500/15の二種類のスーパーコンピュータの導入を決定した。近年，高速大規模の演算のためのスーパーコンピュータは，複数の演算装置(CPU)で同時に処理する並列計算機が実用化され，OSはUNIX系のものが一般的となってきておりジョブはNQS( Network Queuing System)で処理される。

この新スーパーコンピュータの運用を検討する中で，ベクトル計算機VP2600/10Eは，これまでと同じようにMSPで制御しTSSサービスを行なっている汎用計算機M1800とのJES/MAS結合でサービスを行なえるが，UNIXで制御される並列計算機VPP500をどのような形態でサービスするかが課題となった。種々検討した結果，利用者がスムーズに並列計算機VPP500を利用できる環境を提供するには，新たにUNIXのコマンドを覚えて貰うのでは無く，これまでのベクトル計算機と同様にTSSからジョブ依頼できることが必須であると判断し，これを実現する枠組として，M-VPP連携機能[1, 2, 3]を開発した。一方，近年，AVS(Application Visualization System)などUNIXワークステーションのツールを用いてスーパコンピュータの演算結果を可視化するなどの利用も多くなっている。このような利用

では，UNIX の環境から直接 NQS コマンドで VPP500 にジョブ依頼したほうが良い。このために UNIX 環境からバッチ依頼を実現するために利用者管理およびジョブ実行制御機能を開発した。

本稿では，本センターでのコンピュータ VPP500 サービスのために開発した MSP および UNIX からジョブ投入のための機能とその実現方式について報告する。

## 2 システム構成とサービスの概要

### 2.1 VPP500 の構成と機能

本センターが導入した分散メモリ型の並列計算機 VPP500/15 はバッチ処理専用のシステムであり，CP(Control Processor) と 15 台の PE(Processer Element) で構成される VPP と GSP の二つの計算機から構成される (図 1参照)。

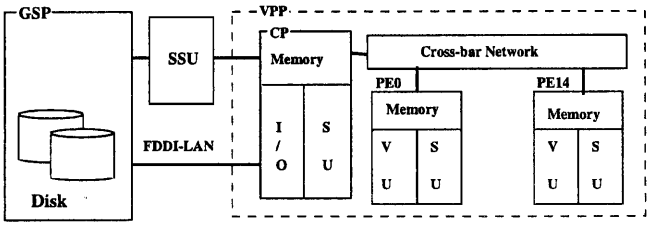


図 1. VPP500/15 のシステム構成

PE は 256MB のメモリと VU(Vector Unit) および SU(Scalar Unit) から構成され，1.6Gflops 処理能力を持つ。各 PE および CP はクロスバーネットワークで結合される。CP と GSP 間は SSU(System Storage Unit) を介して結合されており，VPP のシステムファイルは IPL 時に SSU 上に展開される。一方，GSP は FEP(Front End Processor) であり直接 DISK を持たない VPP に対し SSU あるいは FDDI ネットワークを経由で DISK を提供している。

### 2.2 キューの定義とサービスの概要

本センターでの VPP500 の NQS キュー定義は，表 1のようにしている。キュー名 p,q は，1PE を使用するベクトルジョブ用のもので，並列ジョブには 5PE が使用可能な r と 10PE が使用可能な s を定義している。

表 1. VPP のキュー定義

| キュー名 | PE 数 | 多重度 | CPU 時間 | メモリ量 |
|---|---|---|---|---|
| p | 1 | 5 | 30 分 | 200MB |
| q | 1 | 5 | 180 分 | 200MB |
| r | 5 | 2 | 30 分 | 200MB × 5 |
| s | 10 | 1 | 180 分 | 200MB × 10 |

## 3 MSP からのジョブ投入と制御

### 3.1 システム構成と機能概要

MSP 環境から VPP500 へのジョブ投入を実現する M-VPP 連携機能の設計目標は「MSP と同じ操作で VPP500 にバッチジョブ依頼を可能にする」である。

UNIX と MSP という OS 間でジョブ処理を行なうには，文字コードやデータ形式の変換など色々と課題は多い。しかし，バッチジョブ処理という側面から比較すると基本的な機能は MSP と UNIX のバッチ処理機構 NQS に差はない。

したがって,MSP のジョブ制御文をステップ単位に解釈，NQS のスクリプトを生成，必要なデータ転送を行うことで実現した。図 2に M-VPP 連携のシステム構成とモジュールの機能を示す。

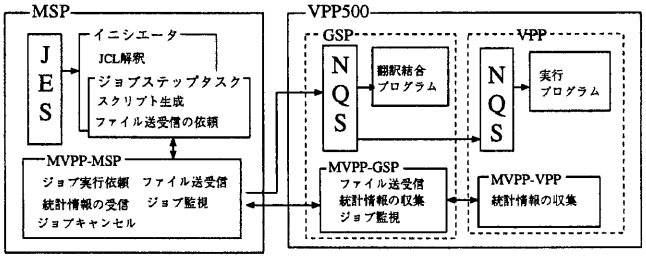


図 2. M-VPP 連携のシステム構成

### 3.2 M-VPP 連携機能と処理

MSP で投入されたジョブは，JES(Job Entry Subsystem) で受け付けられ実行待キューにつながれる。イニシエータは,実行待ちキューからジョブを取りだし制御文の解釈を行ない，必要な資源を割り当て，ジョブステップ単位にジョブステップタスクとしてプログラムを起動する。イニシエータは連携ジョブクラスを処理する場合，M-VPP 連携処理プログラムをジョブステップタスクとして起動する。連携処理プログラムは，制御文の情報をもとに VPP で実行するスクリプトの生成を行ない，また，必要なファイルの転送や実行結果の受信およびスプールへの書き出しを行う。M-VPP 連携機能の実現のために，それぞれのシステムに配置される連携モジュールの役割と機能は，次のようなものである。

**1) MVPP-MSP モジュール**

ジョブ制御のマスターであり連携処理プログラムと協調して必要なファイルの送受信や NQS に対するジョブ依頼および MVPP-GSP,MVPP-VPP モジュールからの統計情報の受信を行なう。また，CANCEL コマンドが投入された場合 NQS へ通知する。

**2) MVPP-GSP モジュール**

VPP 側での連携ジョブ実行を統括するモジュールで MVPP-MSP との間でプログラムやデータの転送を行ない。統計情報の収集と転送および VPP でのジョブの実行を監視を行う。

**3) MVPP-VPP モジュール**

CP に配置されるモジュールで連携ジョブに関する統計情報を収集し，MVPP-GSP に通知する。

## 4 UNIX からのジョブ投入と制御

M-VPP 連携機能で実現した MSP からのジョブ投入は，本センターでの利用者サービスの継続性の保証という独自の課題であるが，UNIX 環境から NQS コマンドを用いてジョブ投入という形態は，VPP 本来のサービス形態であり実現は容易であるといえる。しかし，センターでの運用を考えたとき NQS に対して，次のような機能追加が必要であった。

**1) 利用者当りの投入ジョブ数の管理**

センターでは，利用者が自身でジョブの投入や出力を行うオープンバッチ方式でのサービスを行っている。したがって，特定の利用者が一度に多くのジョブを投入しシステムを占有しないように利用者当りの投入ジョブ数を管理する必要がある。

**2) 同時実行ジョブ数の制御**

投入されたジョブは，基本的に FIFO で処理されるが特定の利用者が連続してジョブを投入した場合，その利用者ジョブでシステムが占有され，他の利用者のジョブが長時間待たされ

てしまう。したがって，実行待ちジョブの取り出しに時に利用者当りの同時実行多重度を制御する必要がある。

**3) 課金のためのコードの指定と課金処理**

センターでは課金を行っているが，課金の支払い費目には校費や科研費など複数のものがあり，ジョブ単位に費目を指定出来る必要がある。また，ジョブ投入時に指定されたコードを課金処理に渡す仕組みが必要である。

また，VPP の pacct ファイルには，M-VPP 連携ジョブと NQS ジョブの課金レコードが混在して出力される。M-VPP 連携機能では，MSP の課金システム連携して課金集計を行っている。したがって，pacct ファイルから NQS ジョブだけのレコードを抽出して，課金処理を行う必要がある。

**4) NQS ジョブと M-VPP 連携ジョブとの混在実行の保証**

ネットワーク構成された NQS システムでは，投入されたジョブは指定されたキューの実行システムまでパイプキュー経由で転送され実行待キューにつながれる。実行待キューのジョブは，FIFO で取り出され実行される。すなわち，NQS ジョブは，パイプキューが停止していない限り投入後直ぐに VPP の実行待キューつながれるのに対して，連携ジョブは，一旦，MSP でスプーリングされステップ単位に NQS ジョブとして投入されるので，連携ジョブが NQS ジョブに比べて実行を待たされる。したがって，連携ジョブと NQS ジョブが同時に混在して実行されるような枠組みが必要である。

これらの機能のなかで 1),2) の利用者ジョブ数の管理および同時実行ジョブ制御は MSP では既に実現された機能である。また，課金データの分離およびジョブの混在実行の保証機能は VPP に対して MSP と UNIX という二つのシステムからのジョブ投入を実現するためには，必須な機能である。

### 4.1 利用者管理機能

利用者管理機能は，利用者当りの投入ジョブ数と課金コードの管理を行う。これは，ジョブの投入コマンドである qsub コマンドに機能追加する形で実装した。

ジョブ数の管理は，qsub コマンドの投入時に NQS にキューイングされている利用者のジョブの数を qstat コマンドを用いて集計しチェックする方式を採った。しかし，この方式では同じ利用者が連続的にジョブを投入した時，qstat コマンドの集計で誤差を生む可能性があり処理の逐次化が必要である。今回の実装では，利用者毎に lock file を作成する事で対処した。

課金コードの指定方法には，環境変数 **(QSUB_ACCT)** と qsub コマンドのオプション **(−A)** の二つがある。両方で指定された場合，コマンドのオプションが有効になる。指定された課金コードは，センターの UNIX 利用者の管理台帳である NIS(Network Information name Service) データベースを検索し，利用期限などの有効性を検査している。また，省略された場合には NIS に定義されるデフォルトの課金コードを使用する。有効性が調べられた課金コードはジョブのリクエスト識別子に埋め込む方式で後の課金処理に渡すようにした。

qsub コマンドに追加した，センター独自の利用者管理ための処理の流れとリクエスト識別子の形式を図 3および図 4に示す。

### 4.2 課金データの分離

UNIX の課金データは，プロセス単位に pacct ファイルに書き出される。M-VPP 連携機能では，MVPP-GSP,MVPP-VPP モジュールが pacct ファイルをアクセスし連携ジョブのレコード抽出して独自に収集している。これらのモジュールに連携ジョブ以外の課金レコードを別ファイルに書き出す処理を追加し，この別ファイルを NQS ジョブの課金に使用することで課金データの分離を行っている。

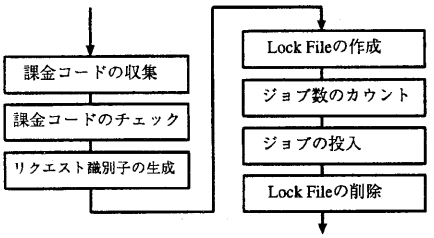


図 3. qsub コマンドの処理


図 4. リクエスト識別子の形式

### 4.3 ジョブの実行制御機能

利用者当りの同時実行ジョブ制御およびNQSジョブと連携ジョブの混在実行ための制御機能は，本来，VPPのNQSの実行ジョブのスケジューリング時に管理すべきである。しかし，VPPのシステムファイルがSSUという揮発性の装置にあり停電事故などでジョブのスプールを失う可能性がある事，また，高価なSSU上にスプーリング用のスペースを大きく確保することは望ましくない。また，VPPへのジョブは，全てパイプキュー経由で転送されるのでNQSのキューイング受付出口で制御する方針を採った。キューイング受付出口とは，ローカルあるいはリモートからのジョブ受付処理時に呼び出されるルーチンで，キューイングの可否をリターンコードで通知できる。リターンコードには，Accept,Reject,Refuseの三つがあり，Acceptはキューイングを可能であり，Rejectはキューイングを拒否し再送を要求する，Refuseはキューイングを拒否し，再送も認めない。

出口での制御は，各キュー毎に定義されるRun Limit(実行多重度)の値を元に，次のような方式で処理を行っている。

**1) キューイング総ジョブ数の管理**

各キューのRun Limit値の2倍の値をシステム内ジョブ数とする。キューイング受付出口では，転送されてきたジョブと同じキューの実行中と実行待ジョブ数の和を求め，これがシステム内ジョブ数を超えてジョブが到着した場合は，キューイングを拒否し再送を要求する(Reject)。システム内ジョブ数以内であれば，キューイングを認める(Accept)。

**2) 利用者の同時実行ジョブ数の管理**

各キューのRun Limit値を利用者当りの同時実行のジョブ数とする。実行中，実行待ジョブ内に到着したジョブの持ち主と同一利用者ジョブが幾つあるかを数えて，これが利用者の同時実行ジョブ数を超えた場合，キューイングを拒否し再送を要求する(Reject)。システム内ジョブ数以内であれば，キューイングを認める(Accept)。

連携ジョブとNQSジョブの混在実行の制御は，連携ジョブの利用者がMSPでジョブを投入した利用者関係ない固有の利用者(mvpp)で処理しており，**1)**でNQSジョブが連携ジョブ比べ先にVPPにキューイングされるのを制御し，**2)**でNQSジョブと連携ジョブが混在して実行されるように制御している。ジョブ実行制御の概念を図5に示す。

**(1)** GSPのNQSが転送待ジョブでQueued状態のものをFIFOで取り出し，まず，ジョブの制御情報をパイプキュー経由でVPPに転送する。転送中のジョブの状態はGSPではRoutingであり，VPPではArrivingとなる。

**(2)** VPPのNQSはパイプキュー経由でジョブのヘッダーを受け取るとキューイング受付出口を呼び出す。

**(3)** キューイング受付出口では，まず，システムのキューイング総数の判定を行い，次に利用者の同時実行ジョブ数の判定を行いキューイングの可否を決定する。

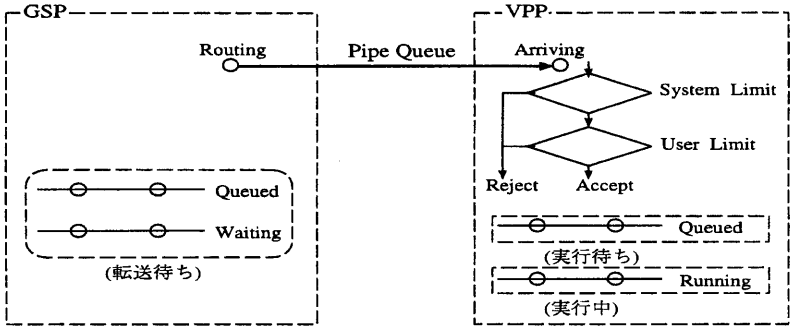

図 5. ジョブ実行制御の概念

**(4)** ジョブのキューイングが認められた場合ジョブのスクリプトが送られが行われ，スプーリングされ実行待キューにつながれ Queued 状態になる。

**(5)** VPP の NQS は，対応するキューの実行中ジョブ数が Run Limit 以下であれば，Queued 状態のジョブを FIFO で取り出し起動しジョブは Running 状態になる。

**(6)** キューイングが拒否され再送要求されたジョブは，転送が中断され，一旦，Waiting 状態に置かれる。Waiting 状態のジョブは一定時間後，Queued 状態になり，再送を待つ。

## 5 まとめ

以上，新スーパコンピュータ VPP500 サービスのために本センターで開発した MSP からのジョブ投入を実現する M-VPP 連携機能および UNIX から直接 NQS コマンドでジョブ投入のために開発した利用者管理とジョブ実行制御の機能と実現方式について報告した。

M-VPP 連携機能の実現は，並列計算機 VPP500 に対しこれまでのベクトル計算機と同じオペレーションでジョブ投入ができるので，利用者に混乱を与えること無くサービスでき当初の目標を達成できた。

また，UNIX からのジョブ投入の実現は，センターの UNIX システムだけでなく研究室のワークステーションとの連携など新たな形態でスーパコンピュータを活用できる基盤が整備出来たと考える。

なお，M-VPP 連携機能の開発は，富士通株式会社との協同で行なわれ，現在，製品として多くの計算センターで導入され，サービスされている。

最後に，NQS からのジョブ投入で色々とアドバイスを頂いた岡部寿男助教授並びに M-VPP 連携機能の検討に加わって頂いた安岡孝一助手をはじめ富士通株式会社の関係者各位に感謝します。

## 参考文献

[1] 金澤正憲, 平野彰雄,「MSP から VPP を利用する」, Scientific System 研究会 Newsletter, No.77, 1995

[2] 金澤正憲, 平野彰雄, 大空 瞭, 山崎 成美,「異種 OS によるジョブの入出力および管理方式について」, 情報処理学会コンピュータシステムシンポジューム論文集,Vol.95,No.7,1995

[3] 金澤正憲, 平野彰雄,「異種 OS 間連携におけるシステム管理方式」, 京都大学大型計算機センター 研究開発部 研究発表報告集 第十一号, 1996