4 システムソフトウェア
― OS，運用管理ソフトウェア，ファイルシステム―

応 専

宇野篤也[*1] 加藤丈治[*2] 宮本巧輝[*2] 岩田章孝[*2] 長屋忠男[*2]

[*1] 理化学研究所 [*2] 富士通（株）

「京」は世界最先端の研究成果を創出するための基盤ツールとして，長時間にわたりトラブルなく稼働できる信頼性を持ち，安定的に稼働し，計算科学分野の幅広いアプリケーションを効率的に実行できるとともに，ユーザにとって使いやすいシステムであることが求められている．

「京」は，8万台を超えるノードから構成されるこれまでにない大規模クラスタシステムである．これほどの規模になると，個々の部品の信頼性を高めることによるシステム全体の信頼性の向上には限界がある．そのため，システムの運用性を高めるために「壊れにくい」「壊れてもすべてが止まらない」「壊れてもすぐ直せる」システムであることが求められ，それらに対応する機能を備える必要がある．また，利便性という面では，本システムを多くの研究者・技術者が同時にかつストレスなく使え，大小さまざまな規模のジョブが互いに影響することなく同時に実行でき，ハードウェアの故障などによるジョブへの影響は最小限にする必要がある．

本稿では，これら「京」に求められる要件を実現するために，システムソフトウェア（オペレーティングシステム，運用管理ソフトウェアおよびファイルシステム）において開発・改良された機能について紹介する．

システムソフトウェア概要

●システムの利用イメージ

「京」では，多数のユーザが多数の計算ノードを使う大規模な計算処理を複数同時に高速に実行する．これらシステムの管理者および利用者が効率的にシステムを管理・利用するための機能はシステムソフトウェアにより提供される．この機能により，システム管理者は，制御ノードから多数の計算ノード・ファイルサーバを一括して起動・停止したり，異常を一元的に監視することができ，利用者はログインノード上でアプリケーションを開発し，数百～数万の計算ノードで大規模なシミュレーションを実行し，大規模な計算データをファイルサーバ上に格納・分析するという一連の作業を効率的に行うことができる（図 -1）．

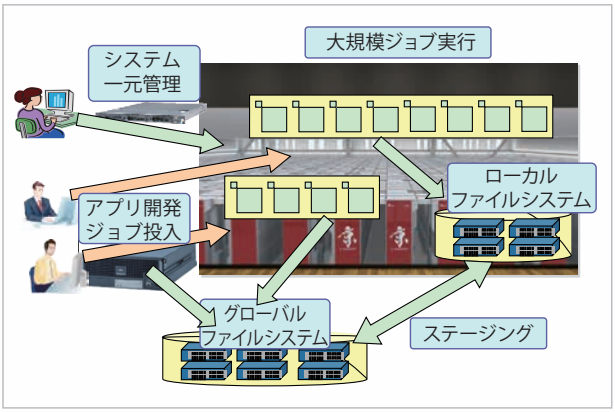

図 -1　システムの利用イメージ

「京」のファイルシステムはジョブの安定した実行を実現するために，ローカルファイルシステムとグローバルファイルシステムという2階層のファイルシステムで構成されている．

ローカルファイルシステムはジョブ専用のテンポラリ領域である．計算ノードと Tofu インターコネクトで接続されている IO ノードに対し，RDMA☆1 通信でデータを volume に書き込むことで高スループットのファイルアクセスを実現している．グローバルファイルシステムはユーザファイルやジョブの入出力ファイルを格納する大容量の領域で，計算ノード群とは別の IA サーバで構成されている．

☆1 Remote Direct Memory Access：オペレーティングシステムを介さずノード間でメモリの内容を転送する機構．

これらのファイルシステム間でのファイル転送を「ステージング」と呼び，ジョブスケジューラと連動してジョブ実行開始に先立ってシステムが自動的に転送する．

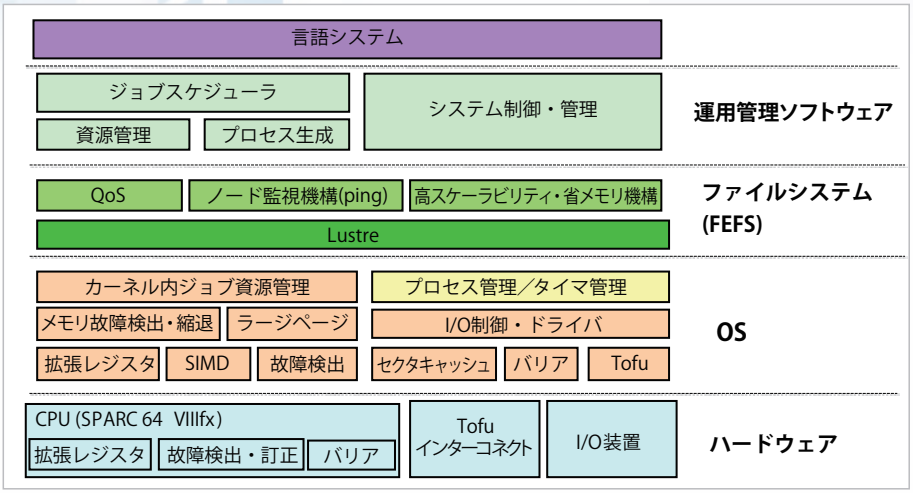

図 -2　システムソフトウェアスタック

●システムソフトウェアの構成

「京」には，並列計算処理を行うアプリケーションの開発／実行／デバッグを行うためのソフトウェアスタックが搭載されている（図 -2）．

言語システム：アプリケーションを開発するためのツール群．開発環境については本特集「5. プログラミング環境」で解説する．

運用管理ソフトウェア：富士通のスーパーコンピュータ技術と経験を元に理研と議論を重ね「京」向けのソフトウェアを設計・開発．

- ジョブ管理：システムの利用効率や利用者の公平利用を考慮し，実行するジョブの要求する資源に応じてスケジューリングを実施．
- システム管理：多数のノードを簡単な操作で制御でき，異常なノードを監視してスケジューリング対象から切り離すなど安定運用のための機能を提供．

ファイルシステム：オープンソースソフトウェアである Lustre をベースに開発．ペタバイトクラスの大規模なデータへの高速アクセスや，多数の利用者で共用できるように容量・性能の管理を行う機能（QoS）を拡張．

OS：Linux をベースに「京」の CPU や Tofu インターコネクトなどのハードウェアをアプリケーションから使用するための機能を拡張．

次章から，OS ／運用管理ソフトウェア／ファイルシステムの特長について述べる．

OS

●「京」における OS の設計観点

「京」の OS は，計算ノードのハードウェア／ソフトウェア性能を引き出すために，以下の 3 点を重視して開発されている．

性能向上機能の搭載

目標性能 10PFLOPS を達成するために，SPARC64 VIIIfx および Tofu インターコネクトのハードウェアを高速にアプリケーションから制御できる手段を提供．

耐故障性の向上

数万の計算ノードから構成される超大規模システムを長期にわたって安定動作させるため，OS レベルでの耐故障性の強化・向上．

既存資産の活用

ユーザの資産を継続利用できるように，TOP500 [☆2] の 91%以上（2011.11 現在）のシステムで使用されている Linux OS の採用．

本章では特に，性能向上と耐故障性の向上に関する OS 拡張機能について述べる．

●性能向上機能

「京」の性能要件を達成するために，OS に対して，以下の 3 つの機能拡張を施している．

ハードウェア拡張直接制御機構

SPARC64 VIIIfx は，SPARC64 VII をベースにレジスタ数拡張，SIMD 命令拡張，ハードウェアコア間バリア機構，セクタキャッシュの搭載などにより数値演算

☆2 世界で最も高速なコンピュータシステムの上位 500 位までを定期的にランク付けし評価するプロジェクト（http://www.top500.org 参照）．

性能を強化した CPU である．

一般的な OS では，デバイスドライバを通して，各種デバイスを操作することで，デバイスに対するアクセスを調停する．しかし，デバイス操作のたびにデバイスドライバの処理によるオーバヘッドが発生するという問題がある．

「京」の OS では，バリア同期やセクタキャッシュなど特に性能が重視されるハードウェアへのアクセスについてはアクセス制御を事前に実施し，アプリケーションから直接制御することで OS によるオーバヘッドが発生しないようにしている．

これにより，システムコールを使用した場合に数 μ 秒かかる処理を数ナノ秒以内に実行できるようになり，アプリケーションの通信性能の向上を実現している．

SPARC64 VIIIfx 拡張レジスタ対応

「京」に搭載されている SPARC64 VIIIfx では，高速な浮動小数点演算を実現するために，64 ビット浮動小数点レジスタを 256 本搭載している．

一般に，浮動小数点レジスタの総サイズは汎用レジスタに比べて巨大であるため，多くの OS では，レジスタの復元をプロセス／スレッドの切り替え時に行わず，アプリケーションが浮動小数点演算を実施する時点まで遅延させる（Lazy FPU[☆3] 機構）．

Lazy FPU 機構は，アプリケーションがレジスタ使用時に OS をトラップして，レジスタの復元を行うため，演算処理実行時に処理コストがかかる．レジスタ退避処理には，μ 秒程度の処理時間を要するため，ナノ秒単位での演算性能が求められる HPC（High Performance Computing）用途では不適切である．

「京」では，レジスタの復元処理をアプリケーション実行前に行うことで，CPU を数値演算処理に集中的に利用可能としている．

ラージページ機構

「京」の OS では，メモリアクセス性能とメモリ使用効率の双方を両立させるための機能として，ラージページ機構を搭載している．

Linux OS では，ハードウェアによる仮想アドレスと物理アドレスの変換のためのキャッシュ（TLB：Translation Lookaside Buffer）を効率的に行うための HugeTLB 機能を搭載している．これはメモリを大きな管理単位（ページサイズ）で管理することでキャッシュできるメモリ範囲を拡大する機能であるが，通常はデータページのサイズを 1 種類しか指定できない．アプリケーションに割り当てるメモリはページサイズで切り上げられるため，ページサイズが大きいほど使用されない領域が増えて，メモリ利用効率が低下するという問題がある．

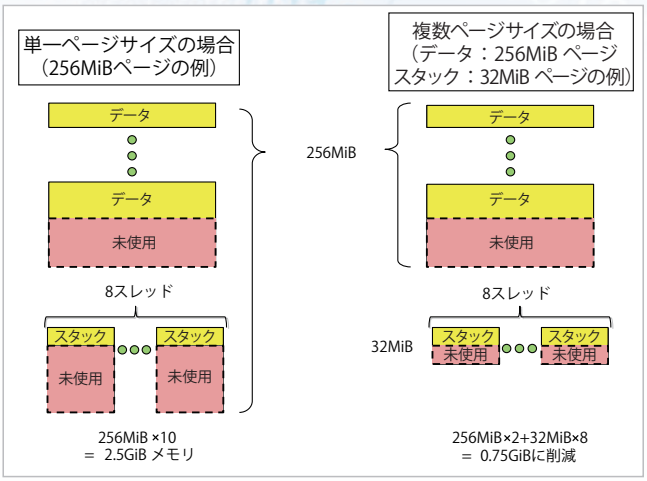

図 -3　複数ラージページによるメモリ効率利用

「京」の OS では，アプリケーション実行時にユーザがページサイズを 4 M バイト／32 M バイト／256 M バイトから選択できるようにし，アプリケーションの特性に応じてページサイズを選択することで，メモリの使用効率と TLB ミス発生回数の軽減を両立している（図 -3）．

●耐故障性向上機能

「京」は，多数のハードウェア部品から構成されるシステムであるため，システム全体での時間あたりの故障頻度は非常に高くなる．そのため，部品の故障時の運用継続や故障個所を早期に検出・交換するための仕組みが必要不可欠である．

「京」は，ハードウェアによる故障メモリ検出機構（メモリパトロール）により，ソフトウェアレベルでのオーバヘッドなしに故障検出を行うことができる．

「京」の耐故障制御機構は，アプリケーションから故障メモリが使用されない限り，アプリケーションの動作を継続させる方針で設計されている．メモリ故障を検出すると，ハードウェアは，OS に故障メモリを通知する．

☆3 Floating Point number processing Unit：浮動小数点演算を専門に行う演算装置．

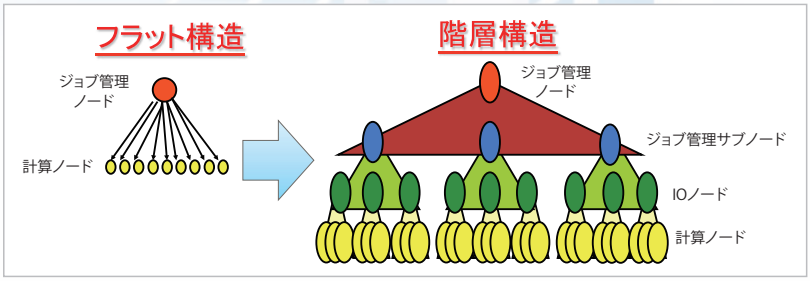

図 -4　階層構造によるノード制御負荷分散

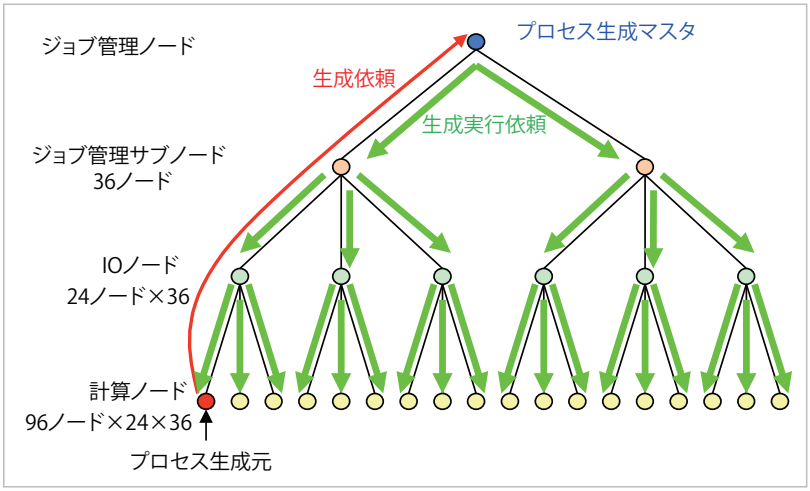

図 -5　階層構造による並列プロセス生成

OS は，ハードから通知された故障個所を利用可能なメモリから除去し，アプリケーションから使用されないようにした上で，運用管理ソフトウェアに故障を通知する．

運用管理ソフトウェアは，OS からの通知に応じて，故障発生ノードの切り離しなどの処理を行う．

運用管理ソフトウェア

●システム管理機能での大規模対応

「京」は 8 万を超える計算ノードで構成される超大規模システムであるが，運用管理の観点からは，「京」を構成する膨大なハードウェア・ソフトウェア群を，あたかも 1 つのシステムとして扱うような操作性が求められる．さらに，一部が故障した状態でもシステム全体としては正常に運用継続できることが重要である．

階層構造による負荷分散

「京」の計算ノード数は膨大なので，従来のようなフラット構造による管理ではジョブ管理ノードがボトルネックとなり，状態監視等に時間がかかることが想定される．また，システム構築段階では月単位で計算ノードが順次増設されていくため，8 万ノードが揃わない段階でも最終的なシステム全体の試験ができることが求められる．このため，IO ノードを含めた 3 階層構造による制御方式を採用した（図 -4）．

この方式では，各階層のノードが 1 段下のノード群だけを管理することで負荷分散を実現している．ソフトウェア開発の試験では，たとえば 1 つのジョブ管理サブノード配下を 1 つのブロックとして動作確認するだけで，大規模な構成と等価な検証が可能となる．

大量プロセス生成の効率化

超大規模並列ジョブを実行する場合，多数の計算ノードで一斉にプロセスを生成するため，そのコストが膨大になる．「京」では，この大量プロセス生成時にも階層構造を利用することで，効率の良いプロセス生成・管理を実現している．具体的には図 -5 に示す階層構造を利用してプロセスを生成する．「京」では，ジョブ管理ノードは 36 台のジョブ管理サブノードと，各ジョブ管理サブノードは 24 台の IO ノードと，各 IO ノードは 96 台の計算ノードとそれぞれ通信するだけで 36 × 24 × 96=82,944 ノードを管理でき，フラット構造に比べ 1 ノードあたりの通信量を大幅に削減することができている．プロセス生成時間を測定したところ，全ノードである 82,944 プロセス生成でも 3.8 秒と非常に高速に生成できていることが確認された．

高可用性の実現

一点故障によるシステム全系停止を防止するため，「京」では，ジョブ管理ノード，ジョブ管理サブノード，IO ノードを冗長構成にしている．トラブル発生時は，自動的にノードを切り替えることでファイルシステムを含めた計算ノード以外のノード故障ではジョブが停止しない設計となっている．計算ノードが故障した場合でも，Tofu インターコネクトは故障ノードを迂回したノード間通信が可能なため，故障ノード以外の計算ノードは引き続きスケジューリング可能である．

●ジョブスケジューラでの大規模対応

「京」のジョブスケジューラには，超大規模並列ジョブ実行（1ジョブ8万プロセス）のみではなく，大量ジョブ（100万オーダ）の同時投入・ノード割り当て・ジョブ実行・ジョブ終了等の処理を高速に安定して動作させることが求められる．

システムスループットの向上

従来の（小規模なシステムの）ジョブスケジューラでは，内部処理をシーケンシャルに実行するタイプも多く，たとえば，大量にジョブ実行を開始しているタイミングではユーザのジョブ投入コマンドが待たされるようなことがある．「京」ではこれまでと比べて計算ノード数，投入ジョブ数が飛躍的に増加することを前提に，ジョブ投入処理，計算ノード割り当て処理，ジョブ実行処理，ジョブ終了処理等，処理ブロックごとに複数のスレッドを用意することで各処理の動作がスケジューラ全体のスループットに影響しない（影響が少ない）設計となっている．

表-1は，実行時間1秒の（瞬時に終了する）ジョブを1万，2万，4万，8万個連続投入する場合のジョブ投入処理，実行開始処理，実行終了処理の1ジョブあたりの各処理時間をまとめたものである．表から分かるように，大量のジョブが実行開始・終了するような高負荷状況においても，ユーザのジョブ投入コマンドが待たされるようなシステム全体のスループットの低下は発生していない．

運用性の向上

運用ポリシーはセンターごとに大きく異なることが多く，ジョブスケジューラにはさまざまな機能が求められる．「京」では，システム内のジョブ実行順序を制御する「スケジューリングポリシー機能」として，グループ／グループ内ユーザ階層を持つフェアシェア・各種優先度・各種制限値等のポリシーを多数用意し，これらのポリシーを評価する順序を自由に設定可能な設計となっている．ジョブが使用できるノード数，経過時間等の制限値データを管理する「ジョブACL[☆4]機能」と組み合わせることで，柔軟なシステム設計およびカスタマイズを可能にしている．

ジョブ数	投入処理	実行開始	終了処理
10,000	1.2ms	30ms	70ms
20,000	1.0ms	30ms	35ms
40,000	1.4ms	22ms	24ms
80,000	2.3ms	23ms	40ms

表-1　高負荷時の1ジョブあたりの処理時間

ファイルシステム

● ファイルシステムの特徴

「京」ではオープンソースのファイルシステムであるLustreをベースにして開発されたFEFS（Fujitsu Exabyte File System）を採用している．大規模システムで動作させるためにFEFSで機能拡張された点について述べる．

多数OSTへの対応

「京」では1ファイルシステム内のOST[☆5]が数千個規模になるため，これまであまり問題ではなかったことが顕著になった．1つはクライアントが各OSTに対して定期的に送っていた監視パケットによるシステムノイズ，もう1つはクライアントの使用メモリ量の肥大化である．

［システムノイズへの対策］

Lustreではクライアントがサーバダウンを検知するためにクライアント側からpingをMDT[☆6]，OST単位で送っていた．そのため「京」では，8万を超えるクライアントから数千規模のOSTに対してpingを定期的に送ることになり，大量の監視パケットがネットワーク上に流れアプリケーションの性能を劣化させていた．FEFSではこの定期的なクライアントからのpingの送信を止め，サーバがリブートもしくはノード交代した際にサーバ側からクライアントに対してpingを送ることでダウンしたことを通知する．

［使用メモリへの対策］

Lustreのクライアントには OST 数に依存してメモリを獲得する個所が多くあり，使用するメモリ量の肥大化によりアプリケーションが使用できるメモリ量やファ

☆4 Job Access Control Lists：ジョブ用のアクセス制御リスト．
☆5 Object Storage Target：ファイルデータの実体を格納する論理volume．
☆6 Metadata Target：メタデータを格納するための論理volume．

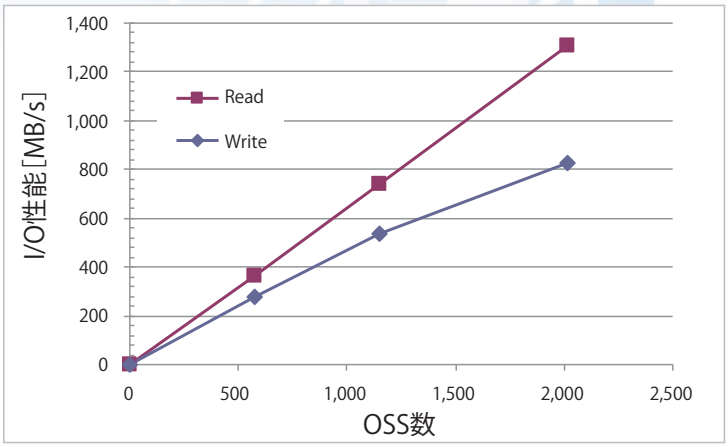

図 -6　ファイル I/O 性能測定結果

イルシステムの性能に影響が出ていた．「京」ではアプリケーションの効率や性能を最適にするため，計算ノードに搭載されたメモリの 90% をアプリケーション専用のメモリとして割り当て，残りの 10% にあたる 1.6GiB でシステムソフトウェアを動作させる必要があり，ファイルシステムが使用するメモリ量を大幅に減らす必要もあった．FEFS では OST ごとに用意されていたロックオブジェクトの管理テーブル，RPC[☆7] リクエストのプール領域，統計情報等を縮小もしくは削除した．また，RPC リクエストの受信バッファサイズも OST 数に依存して大きくなっていたため，クライアント側で最低限必要なサイズのバッファのみを確保し，足りない場合のみクライアント上でメモリを再確保してサーバから RPC リクエストを再送するようにした．これによりファイルがストライプされていない場合には最低限のメモリで動作できるようになっている．その他の対策を含め，70%（約 2GiB）のメモリ量の削減を実現した．

QoS

「京」ではユーザが直接使用するログインノードよりも計算ノードが圧倒的に多いため，ファイルサーバ側で各クライアントからの I/O 要求を均等に処理していたのではログインノードからのレスポンスは悪化する．またログインノード上で特定ユーザが大量の I/O 要求を出したことにより他のユーザのレスポンスが悪くなることがある．こういった問題を解決するため，QoS 機能を実装した．この機能によりログインノードから送られてくる I/O 要求に対してサーバ処理能力を一定の割合で確保したり，1 ユーザが 1 クライアントから同時に発行できる I/O 要求数の上限を制御したりすることが可能となった．

●ファイルシステムの性能

「京」は現在最終調整中であるが，システムの 8 割程度の規模でのファイル I/O 性能を測定した．測定には I/O の代表的なベンチマークである IOR[☆8] を使用した．測定結果を図 -6 に示す．図からも分かるように，Read 性能は 1TB/s を超える 1.3TB/s を達成している．

まとめ

「京」は「壊れにくい」「壊れてもすべてが止まらない」「壊れてもすぐ直せる」システムであることだけでなく，ユーザにとって使いやすいシステムであることが求められている．本稿では，これら「京」に求められている要件を満たすべく開発・改良したシステムソフトウェア（OS，運用管理ソフトウェアおよびファイルシステム）について紹介した．

システムの安定性・運用性やユーザの利便性を高めるため，これら機能の改善・拡張は供用開始後も引き続き進めていく．

（2012 年 4 月 27 日受付）

宇野篤也（正会員）　uno@riken.jp
2000 年筑波大学大学院工学研究科博士課程修了．博士（工学）．現在，（独）理化学研究所 次世代スーパーコンピュータ開発実施本部 開発グループ 開発研究員．システムソフトウェア関連の研究開発に従事．

加藤 丈治（正会員）　kato.takeharu@jp.fujitsu.com
富士通次世代テクニカルコンピューティング開発本部所属．組込みソフトウェア開発を経て，オペレーティングシステムの設計・開発に従事．

宮本巧輝　miyamoto.kouki@jp.fujitsu.com
富士通次世代テクニカルコンピューティング開発本部所属．システムサポート業務を経て，ファイルシステムの設計・開発に従事．

岩田章孝　a.iwata@jp.fujitsu.com
富士通次世代テクニカルコンピューティング開発本部所属．ジョブ管理ソフトウェアの設計・開発に従事．

長屋忠男　nagaya@jp.fujitsu.com
富士通次世代テクニカルコンピューティング開発本部ソフトウェア開発統括部長．

☆7 Remote Procedure Call : 遠隔手続き呼び出し．
☆8 Interleaved-Or-Random : http://sourceforge.net/projects/ior-sio/