# プログラミング環境 EULASH の SNAIL における評価

山本 淳二, 鬼頭 宏幸, 山口 喜弘, 亀井 貴之, 藤原 崇, 米田 卓司, 天野 英晴

慶應義塾大学理工学部

本論文では、容量は大きいが大きなアクセスレイテンシを伴う共有メモリと相対的に少量だが高速にアクセスできるローカルメモリの双方を持つマルチプロセッサのためのプログラミング環境 EULASH をスイッチ結合型マルチプロセッサ SNAIL によって評価を行う。

マルチプロセッサ用ベンチマークプログラム集 SPLASH から 3 つのプログラムを SNAIL に実装することで、EULASH の評価を行った。その結果、EULASH 上の実装ではアドレス変換によるオーバヘッドやバリア同期の実装方法の違いなどからバリア同期が多いプログラムではベアな SNAIL への実装に比べ、実行時間が 2 ~ 4 倍 長くなることがわかった。

また、2 つのスレッドを並行に動作させた場合、単一スレッドの場合に比べて 1.1 倍から 1.4 倍程実行時間が長くなるが、これはコンテキストスイッチのオーバヘッドによると考えられる。

# An evaluation of a programming environment EULASH on the SNAIL

J. Yamamoto, H. kitoh, Y. Yamaguchi, T. Kamei, T. Fujiwara, T. Komeda, H. Amano

Faculty of Science and Technology, Keio University

The EULASH environment is proposed for making the best use of a multiprocessor with a high speed local memory and shared memory with a large latency.

In this paper, the EULASH is evaluated with three benchmark programs from SPLASH on the MIN connected multiprocessor SNAIL. From the result of evaluations, execution times of programs on the EULASH is 2–4 times larger then that of programs without EULASH. The overhead comes from the address translation in the MMU and the inefficient barrier synchronization used in EULASH.

The overhead caused by context switching of threads stretches the execution time from 1.1 to 1.5 times.

# 1 はじめに

ソフトウェア環境 EULASH (Environment for Using Local And SHared memory)[9, 10] は次の 2 つのメモリシステムを持つマシンをターゲットとする。ひとつは高速にアクセスのできるローカルメモリ、もう一つは大きなレイテンシを伴うが全てのプロセッサからアクセスできるリモートメモリである。このタイプのマシンには、複雑なディレクトリキャッシュを持たない単純な NUMA 型システムや、NYU Ultracomputer[1] や BBN Butterfly TC2000[2] のようなスイッチ結合型マルチプロセッサが含まれる。これらの計算機上で高速に実行を行おうとするならば、高速なローカルメモリを効率的に使用することが不可欠である。もしプログラマがプロセスと変数の割り当てを注意深く行うならば共有メモリの使用を最小限にとどめることができる。しかしながら、このことはプログラマに取って大きな負担となる。EULASH は容易なプログラミングとこれらのマシン上での単一ジョブの高速実行を提供するものである。

EULASH では、プログラムは共有変数だけを使用する軽量なスレッドとして記述される。コンパイラは、小規模なデータでプログラムを「試行」することにより、スレッドを仮想プロセッサに割り当て、スレッドのコンテキストスイッチと共有メモリアクセスによるオーバヘッドが最小になるようにプログラムを再構成する。実行時には、高速なスレッドのコンテキストスイッチングと仮想記憶システムを提供するカーネルにより、プログラムを実行する。

これらの機構により、ユーザによる最適化なしに並列アプリケーションプログラムがローカルメモリ、共有メモリの双方を効率的に使用することができる。

本稿では、ネットワーク結合共有メモリ型並列計算機 SNAIL 上で並列ベンチマークを用いて EULASH の評価を行う。

# 2 EULASH の概要

現在までに多くの並列オペレーティングシステムが実装されているが、その多くは主記憶共有型、フロントエンド、マルチユーザを対象としている。これらの OS の多くでは、ローカルメモリを効率良く用いることができず、プロセス生成、スレッドの切替えのコストも大きい。一方、ネットワークで接続された分散システムを念頭に置き、ローカルメモリとプロセス間交信に基づく分散 OS も様々な種類が提案されている。しかし、これらの OS は、共有変数を用いて頻繁に交信するアプリケーションを指向していないため、共有メモリを効率良く用いることができない。

そこで、我々は以下の方針で EULASH を設計した。

- ローカルメモリと共有メモリをそれぞれ最大限に利用する。このために、コンパイラまたはユーザの指示によるデータ配置および mobile 型のデータを導入する。
- コンテキストスイッチングを最大限に高速化する。
- 共有メモリ、ローカルメモリの階層化、仮想化を行なう。
- カーネルは必要最低限の機能のみを持ち、実装・移植が容易な構造を持つ。

これらを実現するために、プログラムをジョブ、プロセス、スレッドの 3 つの処理単位に分けて管理する。

また、カーネルの呼出に伴うオーバヘッドを最小にするために、割り込み処理などカーネルモードでの実行が必要な処理はカーネルで、ユーザレベルで実行できるコードはライブラリとして提供をする。ライブラリとして提供される機能は、ユーザが適宜変更することが可能であるため、例えばロック失敗時にスピンするかコンテキストスイッチするか、アプリケーションに都合の良い方法が選べる。このようにライブラリで多くの処理を提供することでシステムの柔軟性を向上させることが出来る。

# 3 EULASH 上での並列プログラムの実行

EULASH 上で実行されるプログラムは、EULASH のコンパイラを通した後、ジョブ、プロセス、スレッドという処理単位に分割される。

ジョブ ：1 つのアプリケーション・プログラムに相当する処理単位。同一ジョブ内ではグローバルメモリを共有。

プロセス ：個々のプロセッサに割り当てる処理単位。同一プロセス内では全メモリを共有。

スレッド ：実行主体。生成された後は同一のプロセス内で実行を行う。

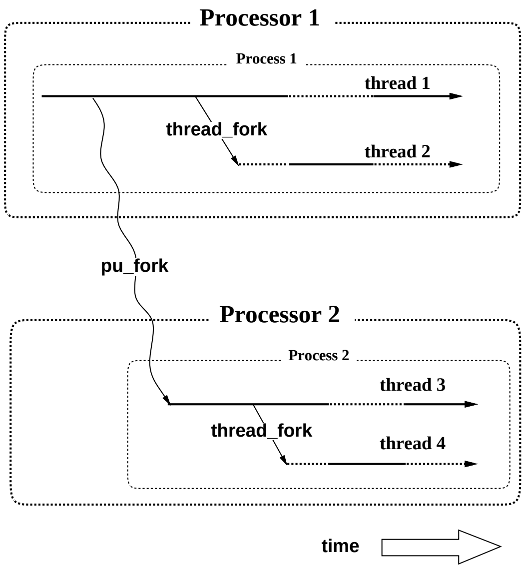


図 1: プロセス、スレッドとプロセッサの関係

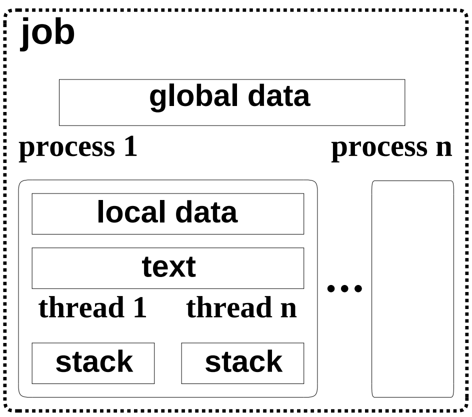


図 2: メモリの共有関係

プロセスは並列に処理される単位であり、仮想プロセッサに対応する。1 プロセッサに 1 プロセスを割り当てる場合が最も効率良く処理を行うことが出来、通常では最大プロセッサ数までのプロセスを生成するようにプログラミングすることを前提とする。また、各プロセスは別プロセッサで実行されるため、プロセス間ではローカルなデータは共有されない。

主記憶共有型の OS では、負荷の軽いプロセッサがレディー・キューに入っているプロセスを実行することが多い。そのため、あるプロセスは休止状態に入る前と次の実行では違うプロセッサに割り当てられ処理される可能性がある。しかし、ローカルメモリを持つマシンをターゲットとした場合、このようなプロセス・マイグレーションを行うと、コンテキストスイッチの度にローカルメモリの内容を移動する必要が生じる。これはローカルメモリ・共有メモリ間のデータ移動であるため非常に高価な処理である。さらに、EULASH では 1 プロセッサにつき 1 プロセスが基本であるため、負荷分散のためにプロセス・マイグレーションは効果がないと考えられる。そのため、 EULASH では、あるプロセスは生成から消滅まで同じプロセッサ上で動作する。

スレッドはプロセスをさらに細かく分けた処理単位である。これはプログラマが処理を記述する上で並行動作として記述する方が良いと思われる時に使用する。同一プロセス上のスレッド (図 1では thread 1 と thread 2 など) は並列に処理されないため、並行動作を行うためにはコンテキストスイッチを必要とする。頻繁なコンテキストスイッチはオーバヘッドをまねくため必要最低限に抑えることが必要となる。そのため、EULASH ではタイマの割り込みによるタイムシェアリングは行わず、ロック獲得失敗時や入出力待ち、ユーザによる明示的な指示が行われた時にのみコンテキストスイッチを行う。 また、ウォッチドッグタイマによるコンテキストスイッチも行う。これはコンテキストスイッチがないために起こるハングアップを防止するためのものである。

# 4 SNAIL

EULASH のターゲットとしているマシンは、コヒーレントキャッシュを持たない大規模 NUMA やスイッチ結合による UMA である。現在、EULASH はスイッチ結合型マルチプロセッサ SNAIL[4] 上で稼働している。

SNAIL (図 3) では 16 プロセッサと 16 共有メモリモジュールが SSS-MIN(Simple Serial Synchronized-Multistage Interconnection Network) と呼ばれる高

速な MIN によって結合されている。プロセッサボードそれぞれに 4 つの CPU (MC68040) とローカルメモリが搭載されている。共有メモリシステムでは Fetch & Decrement と Test & Set のような同期機構がサポートされている。表 1ではメモリシステムに対するアクセス時間を示している。この表からわかるように SNAIL では共有メモリのアクセス時間がローカルメモリに対するそれより 5 倍程大きい。また、MIN が混雑をすると共有メモリのアクセス時間はこの表の値より大きくなり得る。

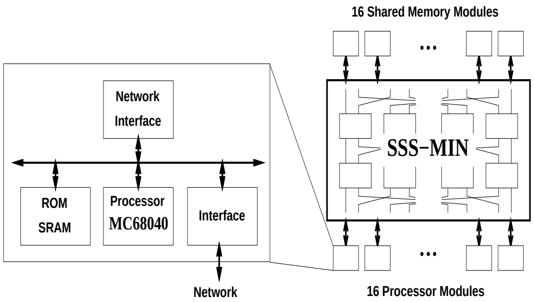


図 3: SNAIL のブロック図

表 1: メモリのアクセス時間 (SNAIL)

| memory system | access type | time(ns) |
|---|---|---|
| local memory | read/write | 250 |
| shared memory | read | 1375 |
| | write | 250 |

# 5 評価

EULASH の評価を共有メモリ型マルチプロセッサシステムのベンチマークプログラム集である S-PLASH ( Stanford ParalleL Applications for SHared-memory ) [6] のうち MP3D, Water, Cholesky を選んで行った。

### MP3D

モンテカルロ法により希薄な流体の流れをシミュレーションするプログラムである。直方体のトンネルのなかに障害物を配置し、そのトンネルを超音速で粒子が通過する際の粒子の動きをシミュレートする。

評価条件:
データファイル test.geom
粒子数 500
ステップ数 50

### Water

このプログラムでは n 個の水分子の液体状態での分子内および分子間の相互作用と水分子のポテンシャルを計算する。

評価条件:
データファイル inputs/sample.in

### Cholesky

このプログラムは、supernodal elimination 法 [7] を用いて疎行列の Cholesky 分解を行なう。Cholesky 分解の ordering, symbolic factorization, numerical factorization の 3 ステップのうち、numerical factorization を並列化して測定対象としている。

評価条件:
データファイル lshp.O

## 5.1 結果および考察

図 4,5,6に各アプリケーションの EULASH 上での実行時間とベアな SNAIL 上での実行時間を示す。

全てのアプリケーションで、ベアな状態に比べ EULASH 上での実行時間のほうが長いが、これはひとつに EULASH カーネルではプロセッサのアドレス変換機構を用いているためであると考えられる。S-NAIL のプロセッサである MC68040 のアドレス変換は MMU (Memory Management Unit) を用いて行われる。この MMU による論理アドレスから物理アドレスへの変換では ATC (Address Translation Cache) にヒットする場合はメモリ参照を行わないが、ヒットしない場合はメモリを 3 回参照する。

また、現在の EULASH 上の実装では、バリア同期は SNAIL の共有メモリに実装されている不可分動作である Fetch&Decrement を用いて実現されている。この実装では、同期の終了を確認するためのリードも必要なため、共有メモリに対して大きな負荷をかける。SNAIL に用いられている SSS-MIN チップはスイッチの優先度が固定であるため、絶対的に優先度の低いプロセッサが存在する。このようなプロ

セッサは優先度の高いプロセッサの同期待ちのリード動作によって、共有メモリを用いた処理を妨げられることになり、実行時間が長くなる傾向がある。

一方、ベアな状態での実装では、同期専用線を用いてバリア同期を実装している。この同期専用線はワイアード・オアのバスで高速に同期を取ることが出来る。さらに共有メモリをアクセスしないため、処理中のプロセッサの動作を妨げることがない。

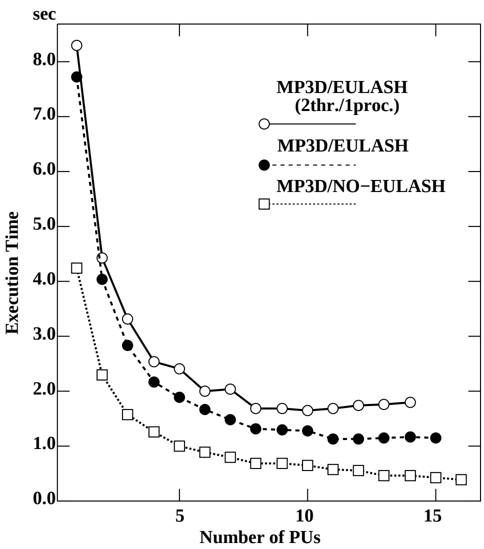


図 4: 実行時間 (MP3D)

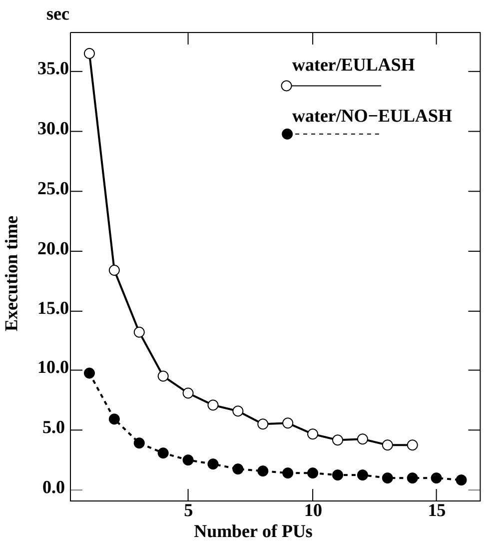


図 5: 実行時間 (water)

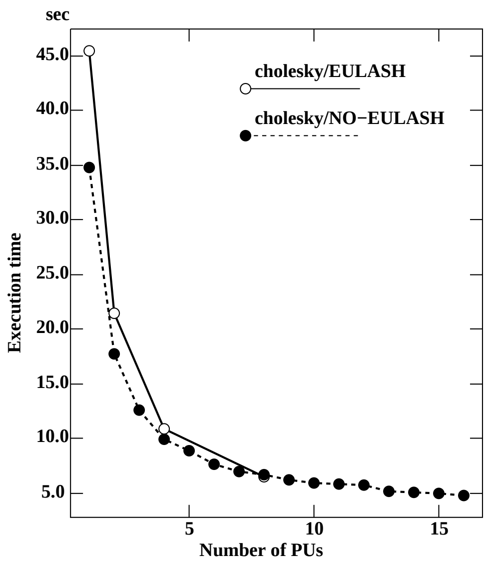


図 6: 実行時間 (cholesky)

このふたつの要因により、EULASH 上での実行はベアな状態での実装に比べ実行時間が MP3D で約 2 倍、water で 3~4 倍長くなっている。

ただし、cholesky(図 6) では、EULASH を使用した場合と、ベアな状態で実行時間にそれほど差がない。これは、cholesky はタスクキューを用いた実装であり、バリア同期は処理の最後に 1 回実行されるだけであるため、このバリア同期の実装による違いが実行時間に反映されにくいためだと考えられる。

複数のスレッドが 1 プロセスに存在する場合、コンテキストスイッチが必要となる。このコンテキストスイッチによる影響を評価するため、MP3D においては 1 プロセスに 2 スレッド生成するプログラムも作成し、実行した (図 4)。各プロセスに 2 スレッド生成した場合、1 プロセスにつき 1 スレッドの場合に比べ、実行時間が長くなるが、これは、スレッドのコンテキストスイッチが必要となるためである。スレッドのコンテキストスイッチでは、プロセッサのレジスタの内容の退避、次に実行するスレッドの選択、レジスタの復帰が必要であるが、この部分は純粋にオーバヘッドである。

# 6　まとめ

共有メモリ型並列計算機のプログラミング環境である EULASH の概要及び SNAIL における実装と評価について述べた。

EULASH はローカルメモリと物理的な共有メモリの両者を持つ並列計算機をターゲットとし、その各メモリの特性に応じたデータ配置を行うことで性能向上を図ることを目的としている。EULASH はバックエンドマシンを対象とし、その処理時間を小さくするように設計を行っている。

SNAIL による評価の結果、現在の EULASH におけるバリア同期の実装は十分に SNAIL の能力を引き出しているとは言えず、高速なハードウエアバリア同期機構である同期専用線を用いたバリア同期の実装が必要である。

現在、システム全体の実装は完了しておらず、そのため各部の検討が十分とはいえない。システムの完成を目指し、実装を完了させ、更に多くのアプリケーションによる評価検討及び改良を行う予定である。

# 参考文献


[1] A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. Mcauliffe, L. Rudolf, and M. Snir, “The NYU Ultracomputer - Designing an MIMD Shared Memory Parallel Computer,” IEEE Transaction on Computer Systems, Vol. c-32, No.2, 1983.

[2] BBN Laboratories, “Butterfly (TM) Parallel Processor Overview,” BBN Computer Company, Cambridge, MA, 1st edition, June 1985.

[3] H. Amano, T. Terasawa, and T. Kudoh, “Cache with synchronization mechanism,” Proceedings of IFIP Congress89, pp.1001–1006, Aug. 1989.

[4] M. Sasahara et al. “SNAIL: A Multiprocessor Based on the Simple Serial Synchronized Multistage Interconnection Network Architecture,” Proceedings of the 1994 International Conference on Parallel Processing, Vol.I, pp.I-117–I-120, Aug. 1994.

[5] H. Amano, L. Zhou, and K. Gaye, “SSS(Simple Serial Synchronized)-MIN: a novel multi stage interconnection architecture for multiprocessors,” Proceedings of the IFIP 12th World Computer Congress, Vol. I, pp.571–577, Sep. 1992.

[6] J. P. Singh, W. Weber, and A. Gupta, “SPLASH: Stanford Parallel Applications for Shared-Memory,” Computer System Laboratory Stanford University, CA 94305, 1992.

[7] E.Rothberg and A.Gupta, “Techniques for improving the performancs of sparse factorization on multiprocessor workstations,” Proceedings of Supercomputing ’90, November, 1990.

[8] 山本 淳二, 徳吉 隆宏, 大和 純一, 服部 大, 天野 英晴, “ローカルメモリを持つ共有メモリ型並列計算機用 OS の設計と実装” 電子情報通信学会技術研究会報告 [コンピュータシステム] CPSY93-35, pp.1–8, 1993 年 11 月.

[9] 山本 淳二, 服部 大, 鬼頭 宏幸, 山口 喜弘, 天野 英晴, “実行前試行によりローカルメモリを持つマルチプロセッサを有効利用する環境: EULASH” 電子情報通信学会技術研究会報告 [アーキテクチャ] ARC95-112, pp.17–24, 1995 年 6 月.

[10] J.Yamamoto, D.Hattori, J.Yamato, T.Tokuyoshi, Y.Yamaguchi, H.Amano, “A preprocessing system of the EULASH: an environment for efficient use of multiprocessors with local memory” Proceedings of the Seventh IASTED/ISMM Intenational Conference, pp.68–71.

[11] 藤原 崇, 鬼頭 宏幸, 山口 喜弘, 塙 敏博, 亀井 貴之, 米田 卓司, 山本 淳二, 天野 英晴, “並列ベンチマークによる SNAIL の評価” 電子情報通信学会技術研究会報告 [コンピュータシステム] CPSY95-99, pp.9-16, 1996 年 1 月.