Vol.2013-ARC-203 No.13
2013/2/1

並列化アプリケーションを対象とした統計的手法による メニーコアアーキテクチャシミュレーションの高速化

阿部洋一[†1]　田口学豊[†1]　木村啓二[†1]　笠原博徳[†1]

本稿では，プログラムのループに着目した統計的サンプリングによるメニーコアアーキテクチャシミュレーションの高速化手法の，自動クラスタリングによるサンプリング位置特定手法を提案する．筆者等による従来の提案手法では，着目するループからサンプリングするイタレーション数を統計的手法によって算出する．さらに，イタレーションごとの実行サイクル数が大きく変化するようなアプリケーションでは，クラスタリングによってプロファイル結果をサンプルサイズが小さくなるように分類することで，低サンプル数で高精度なシミュレーション結果推定を行うことができる．しかしながら，クラスタ数の決定は手動で行う必要があった．本稿ではクラスタリング手法として x-means 法を用いることで，クラスタ数の決定を自動で行う手法を提案する．本手法の予備評価として逐次実行コストの推定を行った結果，最もイタレーション実行コストの変動が激しい MPEG-2 エンコーダの入力動画 SIF16 の場合において, x-measns では 450 イタレーション中の 14 イタレーションをサンプリングすることで 1.92%の誤差が得られることを確認でき，高精度かつ低サンプリング数となるクラスタ数の決定を自動的に得られることが確認できた．

An Acceleration Technique of Many-core Architecture Simulation with Parallelized Applications by Statistical Technique

Yoichi Abe[†1]　Gakuho Taguchi[†1]
Keiji Kimura[†1]　Hironori Kasahara[†1]

This paper proposes an automatic decision technique of the number of clusters and sampling points for an acceleration technique of many-core architecture simulation by statistical methods. This technique, firstly, focuses on a structure of a benchmark program, especially loops. The number of sampling points is exploited from iterations of a target loop by statistical methods. If the variation of the cost of the iterations is large, these iterations are grouped into clusters. Thus, this technique enables higher estimation accuracy with fewer sampling points. However, the number of clusters must be decided by hand in our previous works. The automatic decision technique of the number of clusters by "x-means" is proposed in this paper. As a preliminary evaluation of the proposed technique, sequential execution costs of several benchmark programs are estimated. As a result, when MPEG2 encoder program with SIF16, which causes large variation among the cost of iterations, is used, 1.92% error is achieved with 14 iterations as sampling points of 450 iterations exploited by x-means

†1 早稲田大学
WASEDA UNIVERSITY

1. はじめに

コンピュータアーキテクチャの研究開発では，対象とするアーキテクチャの評価をソフトウェアシミュレーションによって行うことで，さまざまな構成やパラメータの検討を行う．しかしながら，このソフトウェアによるシミュレーションでは，実際のハードウェア上での実行時間と比べて非常に長い時間がかかってしまうという問題点がある．特に，近年普及が進んでいる複数のプロッセッサコアを持つマルチコア，メニーコアアーキテクチャのシミュレーションを行う場合，コア数の増加に応じて実行時間が増大してしまう．

マイクロアーキテクチャシミュレーションの実行時間の短縮という課題については，従来から様々なアプローチが試みられている．例えばシミュレーション対象全体を完全にシミュレーションするのではなく，一部を選択して評価する手法として，SimFlex[1)]や SimPoint[2)]が挙げられる．これらの手法では統計的手法を用いてシミュレーションを行う部分を選択し，高精度でありながら高速にシミュレーションを行うことを可能にしている．しかしながら，これらの手法は基本的にターゲットのアーキテクチャ上では逐次アプリケーションを動作していることを前提としており，マルチコアやメニーコアの特性を十分に活かした並列化アプリケーションを対象とはしていない．すなわち，動機やコア間のリソース競合といった並列化アプリケーションプログラムの構造，及びその実行時の特性を考慮した高速化手法ではなかった．

一方，筆者等はこれまでに，プログラムの大局的な実行コストの推移はプログラムの構造と入力データに依存するという前提の基，並列化アプリケーションのプログラム構造に着目したマルチコア・メニーコアアーキテクチャシミュレーションの高速化手法を提案してきた[3)4)]．本手法では，主としてプログラムの並列化された部分を取り囲むループを統計的サンプリングの対象とし，対象となるループの一部のイタレーションのシミュレーション結果から全体のシミュレーション情報を推定する．

本手法では，まず任意の実機上においてシミュレーションに用いるアプリケーションを逐次実行し，サンプリング対象のループのイタレーションごとの実行クロックサイクル数を計測する．次に，得られた結果から統計的手法を用

いて，サンプルとして詳細に実行するイタレーション数を計算する．シミュレーション時には，シミュレーション精度の異なる形式を実行時に切り替えることにより，並列化アプリケーションにおけるサンプリング対象のイタレーションに対応する部分をサンプルサイズ分だけ詳細なシミュレーションを行い，残りの部分を簡易で高速なシミュレーションを行う．

本手法は，科学技術計算のようなイタレーションごとの実行クロックサイクルの変化が少ない，規則的なアプリケーションをシミュレーションに用いる場合に大きな効果が得られることが示されてきた [3]．

一方，メディア処理を行うアプリケーションなど不規則なアプリケーションの場合，必要なサンプルサイズが大きくなり時間短縮は期待できない．しかしながら，実機でのプロファイリング結果に対して一クラスタ辺りの分散が小さくなるように統計的クラスタリング手法を適用することにより，サンプルサイズを削減しつつ高いシミュレーション精度が得られる可能性を示した [4]．本稿ではさらに，低サンプルサイズで高シミュレーション精度が得られるクラスタ数の自動決定手法を提案し，逐次実行時間の推定を行うことでその予備評価を行う．

以下，2 章ではサンプリングとクラスタリングを用いたシミュレーション高速化手法について，3 章では評価について，最後の 4 章でまとめをそれぞれ述べる．

2. シミュレーション高速化手法

本章では，本稿で提案するシミュレーション高速化手法について説明する．まず 2.1 節でミュレーションの手順の概要を述べ，2.2 節で本手法におけるクラスタリング，2.3 節でサンプルサイズを決定する方法についてそれぞれ説明する．

2.1 高速化手法の概要

本手法ではまず，シミュレーション対象となるマルチコア・メニーコアアーキテクチャを評価するベンチマークプログラムのループ構造に着目する．そして，そのループのいくつかのイタレーションのみを詳細にシミュレーション（サンプリング）し，残りのイタレーションは簡易で高速なシミュレーションにとどめることにより，シミュレーション時間の短縮を図る．

本手法ではベンチマークアプリケーションのループ構造を対象にサンプリングを適用しているため，はじめに，プログラム中のどのループ，あるいはどの階層のループをサンプリング対象とするかを決定する必要がある．本稿では，プログラム中の処理量において大きな割合を占める最外側ループを対象とする．

次に，任意の実機上でシミュレーションに用いるベンチマークアプリケーションを逐次実行し，サンプリング対象ループの 1 イタレーションごとの実行サイクル数を計測する．これによって得た情報をクラスタリングし，統計的手法によってサンプリング対象ループのうちどのイタレーションを詳細に実行するかを決定する．

サンプリング対象イタレーションを決定後，この詳細実行するイタレーション情報を元に，マルチコア・メニーコアシミュレーションを並列化されたベンチマークアプリケーションを用いて行う．ここで用いるシミュレータは，ループのイタレーション単位でシミュレーション精度の切り替えが可能であること，及び 1 イタレーションごとの実行情報を出力することが可能であることを想定する．

最後に，クラスタリング情報とシミュレーション結果を元に，プログラム全体を詳細に実行した場合に得られる結果を推定する．

2.2 プロファイル情報のクラスタリング

本手法ではサンプルサイズを算出する前に，イタレーションごとの実行クロックサイクルを x-means 法[5]によってクラスタリングする．これにより全イタレーションの実行コストから算出されたサンプルサイズの大きな集合を，分散の小さい，すなわちサンプルサイズの小さな集合に分類することで，全体の必要なサンプルサイズを減少させることが期待できる．

x-means 法とは K 平均法(k-means)を発展させたクラスタリング手法である．非階層的クラスタリング手法である K 平均法は，クラスタ数 k が未定の集合に対して，分析を行う者が適宜値を設定しなければならないという問題があった．x-means 法は，まず小さなクラスタ数 k_0（特に指定しなければ 2）による k_0-means を行う．さらに分割後の各集合に対して，分割が適当でないと判断されるまで，k-means を再帰的に行う．

k-means 法を繰り返すことによって，得られる分割数が不定(x)であるために x-means と呼ばれる．この方法を用いることで，入力集合に対してクラスタ数を検討する必要がなくなり，シミュレーション高速化全体のプロセスを自動化することが容易になる．なお，本稿の評価におけるクラスタリングでは，分割停止基準としてサンプルサイズを用いる．分割のプロセスが異なるため，ある集合 C に対して x-means 法を行った結果得られたクラスタ数 k_xによって，通常の K 平均法を C に行った場合，クラスタ数は同じでもそれぞれ結果として得られる集合は異なる．

実機上の実行におけるループイタレーションを，その順序通りシミュレータ上の実行時のイタレーションと 1 対 1 対応させ，クラスタリング結果はそのまま反映できるとみなす．こうして x-means 法によるクラスタリング結果により，ループ中のサンプリング位置を特定する．

2.3 サンプルサイズ決定手法

2.2 節で述べた x-means 法による実機上のイタレーションごとの実行コストのクラスタリング後，各クラスタのサンプルサイズを決定する．

本手法におけるサンプルサイズとは，サンプリング対象ループの全てのイタレーション（あるいはクラスタリング後の各クラスタに属する全てのイタレーション）を母集団とし，そのうち詳細にシミュレーションを行う必要のあるイタレーションの数である．サンプルサイズは，推定する全体の実行クロックサイクル数が期待する誤差に収まるように統計的手法によって算出する．この計算には，各集合の相加平均 μ，標準偏差 σ と，標準正規分布の上側 P%点，許容する誤差率を用いる．本稿では標準正規分布の上側 P%点は P=2.5 の時の 1.96 を，許容する誤差は 5%，すなわち 0.05 をそれぞれ用いるとする．

クラスタリングによって得られた集合を $C_1, C_2,\ldots, C_k$ とする．i=1, 2,…, k について，以下の(1)式によってサンプルサイズ n_i を計算する．

$$\text{サンプルサイズ} n_i \geq \left(\frac{1.96}{0.05} \times \frac{\text{標準偏差}\sigma_i}{\text{相加平均}\mu_i}\right)^2 \quad (1)$$

ただしクラスタリングによって要素数が 1 となった集合については以下の(2)式とする．

$$\text{サンプルサイズ} n_i = 1 \quad (2)$$

シミュレーション後，C_i(i=1, 2,…, k)について推定シミュレーションサイクル数を計算する．

$$\cos t_i = C_i\text{の詳細シミュレーションサイクル数} \times \frac{C_i\text{の要素数}}{n_{ii}} \quad (3)$$

全体の推定シミュレーションサイクル数は，以下(4)式によって得られる．

$$COST = \sum_{i=1}^{k} \cos t_i \quad (4)$$

3. 評価

本章では，本稿で提案する手法に関する評価を行う．まず，評価に用いるアプリケーションについて説明する．次に，実際のサーバー上において評価アプリケーションを実行し，サンプリング対象ループの 1 イタレーションごとの実行クロックサイクル数を計測する．得られたデータに対して K 平均法(k=1, 2, 4, 8, 16, 32，k=1 についてはクラスタリングを行わないことと同義)，x-means 法によるクラスタリングを行い，2 章で説明した手法によって必要なサンプルサイズを求め，各クラスタの n_i を合計したもの比較する．また，実機上の実行で得られた情報をサンプリングの対象とみなして推定誤差の計算を行い，比較する．

3.1 評価アプリケーション

3.1.1 art

art は SPEC CPU 2000 に含まれるベンチマークアプリケーションの 1 つであり，ニューラルネットワークを用いて画像認識を行うアプリケーションである．

プログラム中で大きな割合を占めるのが train_match 関数と match 関数であり，それぞれ呼び出し先でループを持つ．これら 2 つの関数内のループをサンプリング対象とする．なお，今回用いた標準的な実行条件においては，train_match 関数のループが 554 回転，match 関数のループが 4700 回転である．

3.1.2 equake

equake も同様に，SPEC CPU 2000 に含まれるベンチマークアプリケーションの 1 つである．メッシュ状にモデリング化した地形を伝わる地震波の影響をシミュレーションするプログラムである．

処理の大半を占めている大きな for ループサンプリング対象とする．このループは，標準的な実行条件の場合 3855 回転である．

3.1.3 MPEG-2 エンコーダ

MPEG-2 エンコーダは MPEG2 の規格に沿った動画像圧縮処理を行う MediaBench に含まれるプログラムである．入力動画は 1 フレームずつ順番に処理され，この処理の 1 単位はピクチャと呼ばれる．入力フレーム列は I, P, B の 3 種類のピクチャタイプの規則的な並びとして扱われる．図 1 に，その並び方の一部を示す．また，図 2 に 1 ピクチャに対して行われる処理の流れをそれぞれ示す．ピクチャタイプによって，適用される処理の内容が一部異なるため，ピクチャタイプの違いは計算量の違いに大きく表れる．本評価では，この 1 ピクチャ単位のループをサンプリング対象とする．

メディアアプリケーションは，入力データの違いが計算量に大きく影響を与えることが多い．今回の評価については，NHK システム評価用標準動画像解析度 SIF(352x240)[6] から 2 種類の動画を選択して，各々タイトル部分を除いた 450 フレーム分について評価を行った．1 つは定点カメラによる撮影で動きの小さい No.06 交差点である．もう 1 つは水族館のショーでジャンプするシャチを追いかける，比較的動きの大きな No.16 シャチのジャンプである．

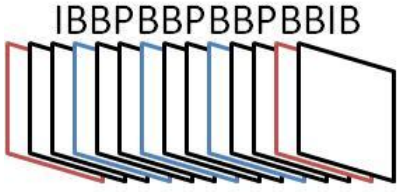

図 1　MPEG-2 エンコーダのピクチャタイプの並び

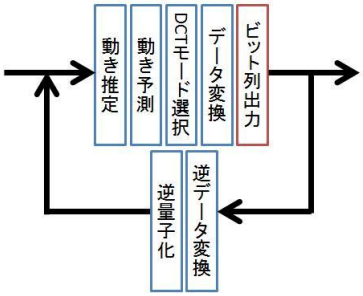

図 2　MPEG-2 エンコーダの 1 フレームに対する処理

3.2 評価環境

今回の評価に当たって，各アプリケーションのコンパイルは gcc version 4.3.3 –O2 によって行った．また，表 1 に今回の評価を行う実サーバーの仕様を示す．

表 1　評価用サーバーの仕様

CPU	Intel Xeon X5670
CPU Clock	2.93GHz
L2 Cache	12MB / core
OS	Debian GNU/Linux 6.0.6

3.3 評価結果

3.3.1 art

train_match 関数の外側ループ 1 イタレーションごとの実行サイクル数を図 3 に示す．match 関数の外側ループ 1 イタレーションごとの実行サイクル数をに図 4 に示す．図 3, 図 4 の結果に対して x-means 法を行った結果，train_match 関数はクラスタ数 4，match 関数はクラスタ数 13 に分類された．

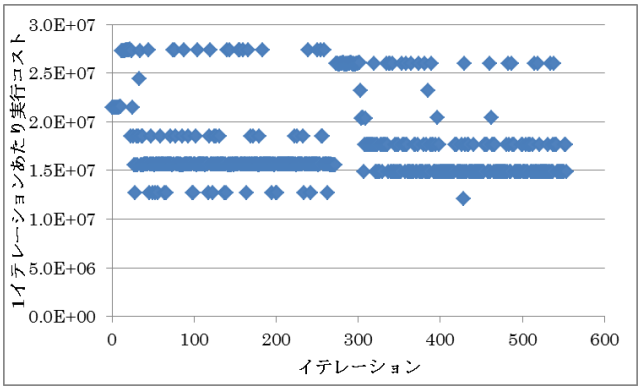

図 3　art - train_match 関数の実行コストの変化

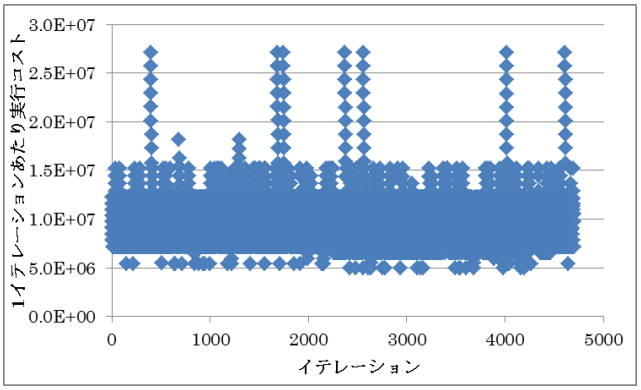

図 4　art – match 関数の実行コストの変化

K 平均法におけるクラスタ数を変化させたときのサンプルサイズの違いを図 5 に示す. 横軸は K 平均法におけるクラスタ数，ただし一番右 x4-13 は x-means の場合の結果である．同様に，クラスタ数を変化させたときの推定誤差の違いを図 6 に示す.

図 5 より，K 平均法では train_match においてクラスタ数が 8 の時に最小のサンプルサイズ 11 を，match においてクラスタ数 16 の時に最小サンプルサイズ 43 を得られたことがそれぞれわかる．同様に x-means では train_match においてサンプルサイズ 11，match においてサンプルサイズ 31 を得ている．

推定誤差に関しては，図 6 よりクラスタ数 2 の時に train_match において最小誤差 0.27%，match においてクラスタ数 32 で最小誤差 0.12%を得ている．先のサンプルサイズ最小のクラスタ数で比較すると，train_match においてサンプルサイズ最小であったクラスタ数 8 の時は誤差 1.34%，match においてサンプルサイズ最小であったクラスタ数 16 の時は誤差 0.20%となっている．

その一方, x-means では train_match において誤差 0.24%，match において誤差 0.01%を得ている．

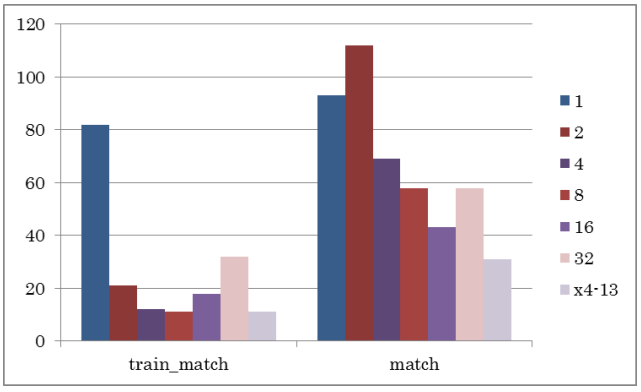

図 5　サンプリングサイズの比較(art)

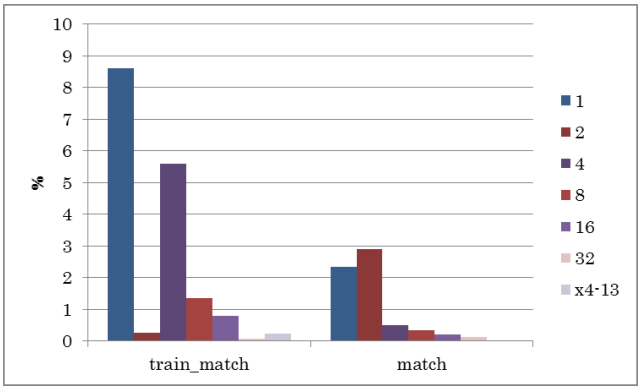

図 6 推定誤差の比較(art)

3.3.2 equake

サンプリング対象ループ 1 イタレーションごとの実行サイクル数を図 7 に示す．図の結果に対して x-means 法を行った結果，クラスタ数 5 に分類された．

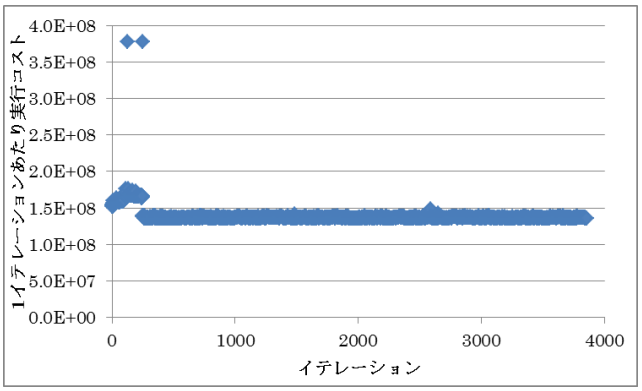

図 7 equake のメインループ実行コストの変化

K 平均法におけるクラスタ数を変化させたときのサンプルサイズの違いを図 8 に示す. 横軸は K 平均法におけるクラスタ数, ただし一番右の x5 は x-means の場合の結果である．同様に，クラスタ数を変化させたときの推定誤差の違いを図 9 に示す．

図 8 より，K 平均法ではクラスタ数が 4 の時に最小のサンプルサイズ 5 を得た．同様に x-means においてもサンプルサイズ 5 を得ている．

推定誤差に関しては，図 9 よりクラスタ数 4 の時に最小誤差 0.01%を得ている．これは先のサンプルサイズ最小の時のクラスタ数と一致している．一方で x-means では誤差 0.41%となっている．

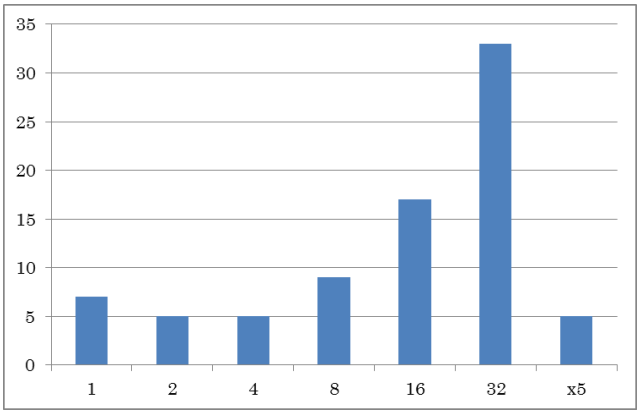

図 8 サンプリングサイズの比較(equake)

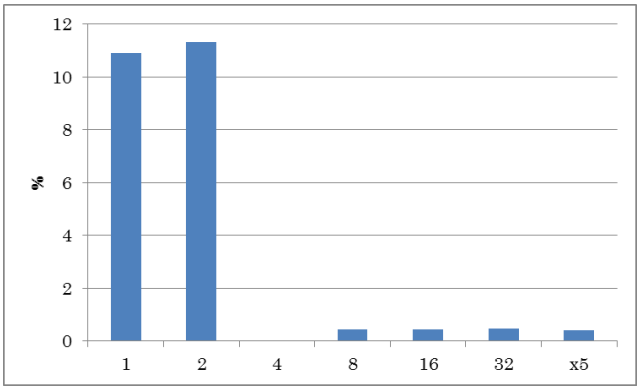

図 9 推定誤差の比較(equake)

3.3.3 MPEG-2 エンコーダ

サンプリング対象ループ 1 イタレーションごとの実行サイクル数を図に示す．図の結果に対して x-means 法を行った結果，SIF06 はクラスタ数 4，SIF16 はクラスタ数 7 に分類された．

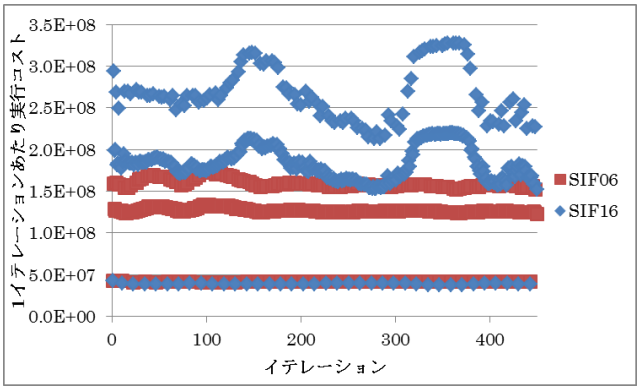

図 10 MPEG-2 エンコーダフレーム単位の実行コストの変化

K 平均法におけるクラスタ数を変化させたときのサンプルサイズの違いを図 11 に示す．横軸は K 平均法におけるクラスタ数，ただし一番右の x4-7 は x-means の場合の結果である．同様に，クラスタ数を変化させたときの推定誤差

の違いを図 12 に示す.

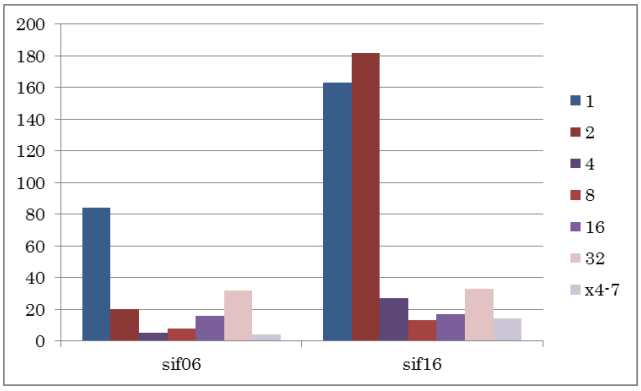

図 11　サンプルサイズの比較（MPEG-2 エンコーダ）

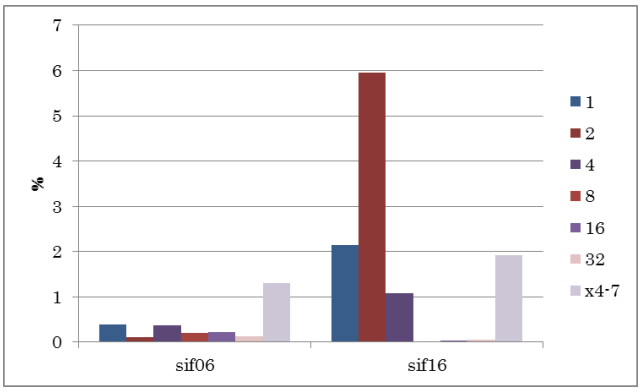

図 12　推定誤差の比較（MPEG-2 エンコーダ）

図 11 より, K 平均法では SIF06 においてクラスタ数が 4 の時に最小のサンプルサイズ 5 を，SIF16 においてクラスタ数 8 の時に最小サンプルサイズ 13 を得られたことがそれぞれわかる．同様に x-means では SIF06 においてサンプルサイズ 4，SIF16 においてサンプルサイズ 14 を得ている．

推定誤差に関しては，図 12 より SIF06 においてクラスタ数 2 の時に最小誤差 0.10%，SIF16 においてクラスタ数 8 で最小誤差 0.00%を得ている．SIF16 においては先のサンプルサイズ最小のクラスタ数と同じである．SIF06 においてサンプル数最小であったクラスタ数 4 の時は誤差 0.37%となる. 一方, x-means では SIF06 において誤差 1.30%, SIF16 において誤差 1.92%を得ている.

3.4 考察

今回，クラスタ数を徐々に大きくしながら K 平均法を行い，得られるサンプリングサイズを比較するということを行った. いずれの結果からも, ある程度のクラスタ数（4~20 程度）を与えると必要なサンプルサイズが大きく減少していることがわかる．その一方で，クラスタ数を増やしすぎると却ってサンプルサイズが増えてしまっている例も存在している．これは(4)式の制約により，各クラスタで少なくとも一度はサンプリングする必要があるためである．つまり，サンプルサイズを小さくするクラスタ数には上限がある.

一方，x-means によって得られるサンプルサイズは，K 平均法において少ないサンプルサイズが得られるクラスタ数に近い値が得られている.

誤差の比較においては対象によって様々な分布を見せているが，全体として第 2 章で想定した許容する誤差である 5%を大きく下回っているものが多い．特に，サンプルサイズが小さくなるようなクラスタ数 k の K 平均法や，x-means の場合においてこの目標値は達成されており，最もイタレーション実行コストの変動が激しい MPEG-2 エンコーダの入力動画 SIF16 の場合において，K 平均法では 0.00%の誤差，x-means では 1.92%の誤差となった．また，図 8，図 9 の equake の結果から，クラスタリングによってサンプルサイズがほとんど改善しない場合でも，推定誤差の改善が見込めることがわかる.

以上のことから，本手法における自動クラスタリング手法として x-means を用いることは妥当であると考えられる.

4. まとめ

プログラムのループ構造に着目した統計的サンプリングによるシミュレーションの高速化について，クラスタリングによってサンプリング位置を特定することで，分散が大きくサンプルサイズが増大してしまうアプリケーションを用いる場合の実行時間を改善できる見込みを示した.

また，その際クラスタリング手法として x-means 法を用いることを提案した.

本手法の予備評価として，逐次実行コストの推定を行った結果，最もイタレーション実行コストの変動が激しい MPEG2 エンコーダの入力動画 SIF16 の場合において，x-measn では 450 イタレーション中の 14 イタレーションをサンプリングすることで 1.92%の誤差が得られることを確認でき，高精度かつ低サンプリング数となるクラスタ数の決定を自動的に得られることを示すことができた.

謝辞　本研究の一部は科研費若手研究（B）23700064 の助成及び，経産省グリーンコンピューティングシステム研究開発により行われた.

参考文献

1) Thomas F. Wenishch, Roland E. Wunderlich, Michael Ferdman, Anastassia Ailamaki,　Bavak Falsafi, and James C. Hoe, “Sim-Flex: Statistical Sampling of Computer System Simulation”　Micro IEEE, Volume 26, Issue 4, pp.32-42, July-Aug, 2006
2) Erez PerelmanGreg HamerlyMichael Van Biesbrouck Timothy SherwoodBrad Calder　“Using SimPoint for Accurate and Efficient Simulation”　SIGMETRICS ’　03, San Diego, California, USA. ACM 1-58113-664-1/03/0006, June 10—14,

2003
3) 石塚亮，阿部洋一，大胡亮太，木村啓二，笠原博徳，“科学技術計算プログラムの構造を利用したメニーコアアーキテクチャシミュレーション高速化手法の評価”, 情報処理学会研究報告．計算機アーキテクチャ研究会報告
2011-ARC-196(14), 1-11, 2011-07-20
4) 阿部洋一，石塚亮，大胡亮太，田口学豊，木村啓二，笠原博徳, "並列化メディアアプリケーションを対象としたメニーコアアーキテクチャシミュレーションの高速化の検討"，情報処理学会第 191 回計算機アーキテクチャ研究会, Vol. 2012-ARC-199(3), 1-4, 2012-03-27
5) 石岡恒憲, “x-means 法改良の一提案―k-means 法の逐次繰り返しとクラスターの再結合―”，計算機統計学 18(1), 3-13, 2006-06-30
6) 財団法人 NHK エンジニアリングサービス