CBE を用いた AES-CTR モードの実装

杉浦 寛† 大釜 正裕† 黒羽 秀一†† 齋藤 孝道†
† 明治大学 †† 明治大学大学院

1 はじめに

一つのプロセッサ上に複数のコアを搭載する技術の発達により，マルチコアプロセッサが普及してきている．マルチコアプロセッサは一つのプロセッサとして動作するが，内部では，複数のコアがそれぞれ他のコアに依存せずに並列に動作することによって，プロセッサ全体としての性能向上を図っている．そのため，既存のソフトウェアをマルチコアプロセッサに対応させることにより，様々なケースにおいて処理の高速化が期待できる．

現在利用されているマルチコアプロセッサの一つに，Cell Broadband Engine (以降 CBE と呼ぶ) [1][2] がある．CBE は，異なる種類のコアを複数搭載したヘテロジニアス・マルチコアプロセッサであり，OS などの制御を受け持つ汎用コアである PPE (PowerPC Processor Element) を 1 基と，高速な処理を実現するための演算用コアである SPE (Synergistic Processor Element) を 8 基搭載している[1]．

本論文では，暗号処理の高速化を目的として，この SPE に共通鍵暗号化方式のブロック暗号アルゴリズムである AES の CTR モードを実装し，暗号処理の並列化を試みた．そのパフォーマンス計測を行い，結果について考察する．

2 実装環境

2.1 CBE 開発環境

本論文では，開発環境として PLAYSTATION3 に搭載されている CBE を使用した．また，CBE 上での開発を行うにあたって，FedoraCore6 (Kernel2.6.16) を，CBE 用のカーネルパッチを当てて使用した．さらに，FedoraCore6 上で CBE の開発を行うために，CBE 開発者向けのツールキットである CellSDK 2.1[3] を使用した．この環境下で，PPE 及び SPE をターゲットとしたコンパイラである ppu-gcc4.1.1, spu-gcc4.1.1 を用いて実装を行った．

2.2 実装内容

ここでは，CBE 上での AES-CTR モードの実装について示す．本論文では，AES-CTR モードの C 言語コード [4] (以降 AES-CTR コードと呼ぶ) を用いた．このコードを改変し，SPE 上で実行するコード (以降 SPE コードと呼ぶ) で暗号処理を行い，PPE 上で実行するコード (以降 PPE コードと呼ぶ) でその制御を行うよう実装した (以降実装コードと呼ぶ)．実装では SPE を 6 基用い，各 SPE でそれぞれ同じ SPE コードを実行した．ただし，使用する SPE の数は指定できるようになっている．図 1 は，処理を図示したものである．以下に示す説明文に割り当てた番号は，図 1 の各処理に割り当てたそれと対応している．また，簡略化のために SPE コードは一つだけ表記している:

(1) SPE の初期化
PPE コードは，SPE コンテキストの生成，実行コードの LS[2] へのロードを行う．SPE コンテキストとは，SPE の状態を表すものであり，これを用いて SPE コードの実行や終了を指示する．

(2) データ/処理の分割
暗号処理に入力として与える平文とカウンタ，暗号処理に用いる鍵を用意する．平文は，各 SPE に渡すために分割する．

(3) SPE コードの呼び出し
PPE コードがスレッドを生成し，生成したスレッドから，SPE コードを呼び出す．SPE コードを実行する際に，分割した平文とカウンタおよび鍵のアドレスを引数として渡す．

(4) SPE コードでのデータ処理
SPE コードは，DMA 転送によってメインメモリから LS へ平文データを転送する．全ての SPE でのデータ転送が終了すると，(5) の処理で，一斉に暗号処理を開始する．SPE での暗号処理が終了すると，LS からメインメモリへ処理結果をデータ転送し，SPE における処理を終了する．

(5) SPE 制御関数の呼び出し
PPE コードでは，スレッドを生成した後に SPE 制御関数を呼び出し，SPE との同期をとる．

(6) SPE コードの終了待ち
SPE コードを実行しているスレッドは，全ての SPE コードが終了すると，スレッドを破棄する．

(7) 処理の終了
全てのスレッドが破棄されると，PPE における処理を終了する．

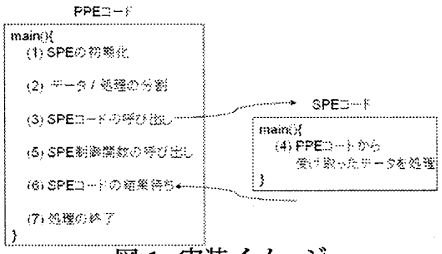

図 1: 実装イメージ

3 評価

3.1 計測環境

実装コードの評価をするために，暗号処理に入力として与えるデータを 16byte，64byte，256byte，1024byte ずつ暗号化した場合のスループットを計測した．また，実装コードとの比較のため，オリジナルの AES-CTR コードと OpenSSL speed コマンドを用いた．加えて，CBE 環境との比較のため，Pentium4 環境でも計測を行った．計測は 10 回行い，その平均値を結果とした．

(A) CBE 環境
PPE と SPE は共に 3.2GHz で動作し，CBE 自体は 256MB の RAM を搭載している．パフォーマンス計測を行うのは，実装コードと，AES-CTR コード，OpenSSL-

† Kan SUGIURA, ,Masahiro OHGAMA, Takamichi SAITO
†† Shuichi KUROBA
{kan_s, ohgama, kuroba, saito}@cs.meiji.ac.jp
Meiji University(†), Graduate School of Meiji University(††)
1-1-1 Higashimita, Tama-ku, Kawasaki-shi, Kanagawa, 214-8571, Japan(†)(††)

[1] ただし，8 基の SPE のうち，2 基は使用できないように設定されているので，実質的に使用できるのは 6 基である．

[2] Local Storage. 各 SPE が持つ 256KB のメモリである．

0.9.8b[5] の speed コマンドである．OpenSSL では，AES-CTR モードの API は提供されているが，speed コマンドには含まれていない[3]ので，今回の計測用に AES-CTR モードを適用修正した．ただし，SPE 上では OpenSSL は動作することができないため，計測を行わない．以下に各プロセッサコアとパフォーマンス計測を行うコードとの関係を示す．

- SPE 6 基を用いた場合

  SPE を 6 基用いて，実装コードを実行し，暗号処理のスループットを計測した．計測は，図 2 中の処理時間 t に示すように，PPE コードが暗号処理開始を通知した時点から，全ての SPE コードから暗号処理終了を受け取った時点までとした．図 2 中の実線が PPE コードの処理を，破線が SPE コードの処理を示しており，spe_1〜spe_6 は SPE コードの実行開始をそれぞれ表している．暗号処理に入力として与えるデータサイズの合計は 600KB とし，それを 100KB に分割して，各 SPE に渡している．

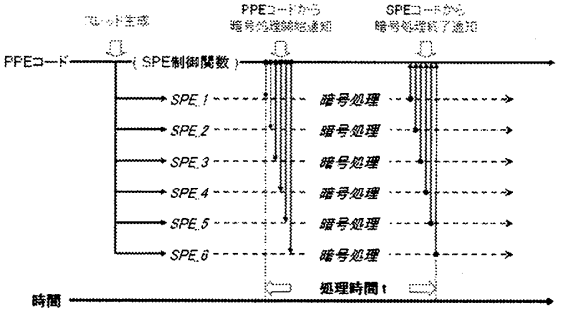

図 2: SPE を用いた暗号処理の計測イメージ

- SPE 1 基を用いた場合

  SPE を 1 基のみ用いて，実装コードを実行し，暗号処理のスループットを計測した．計測は，計測開始時点及び計測終了時点を SPE 6 基と同様とした．暗号処理に入力として与えるデータサイズの合計は，LS のサイズ制限により，100KB とした．

- PPE を用いた場合

  PPE のみを用いて，AES-CTR コード及び OpenSSL speed コマンドを実行し，暗号処理のスループットを計測した．AES-CTR コードの計測は，AES-CTR モードの処理を開始した時点から処理が終了した時点までとした．暗号処理に入力として与えるデータサイズの合計は 600KB とした．

(B) Pentium4 環境

Pentium4 (3.2GHz) で 512MB の RAM を搭載したマシンであり，FedoraCore6 (Kernel2.6.16) が稼動している．パフォーマンス計測を行うのは，AES-CTR コードと OpenSSL speed コマンドである．AES-CTR コードの計測は，計測開始時点と計測終了時点を PPE と同様とした．暗号処理に入力として与えるデータサイズの合計は 600KB とした．

3.2 AES-CTR コードの処理時間の計測

AES-CTR コードによる暗号処理の処理時間の計測を行った．表中の Block size は，暗号処理に入力として与えたデータを分割する単位で，一度に暗号化を行うデータサイズを表している．PPE と Pentium4 上で実行した場合の計測結果を表 1 に示す．

3.3 OpenSSL speed コマンドの処理時間の計測

OpenSSL speed コマンドによる暗号処理の処理時間の計測を行った．PPE と Pentium4 上で実行した場合の計測結果を表 2 に示す．

3.4 実装コードの処理時間の計測

実装コードによる暗号処理の処理時間の計測を行った．SPE を 1 基または 6 基用いた場合の計測結果を表 3 に示す．

表 1: AES-CTR コードの処理時間

Block size (byte)	16	64	256	1024
PPE (MBps)	17.49	16.54	17.37	18.20
Pentium4 (MBps)	61.74	57.67	56.44	58.17

表 2: OpenSSL speed コマンドの処理時間

Block size (byte)	16	64	256	1024
PPE (MBps)	31.08	33.62	33.15	32.51
Pentium4 (MBps)	71.07	74.89	75.83	76.17

表 3: 実装コードの処理時間

Block size (byte)	16	64	256	1024
SPE*6 (MBps)	168.77	161.66	170.50	177.40
SPE*1 (MBps)	33.76	32.94	34.48	34.98

4 考察

ここでは，CBE 環境と Pentium4 環境における AES-CTR モードのスループットを比較することで，そのパフォーマンスについて考察する．

今回の実験では，SPE を 6 基用いて実装コードを実行し，1024byte ずつ暗号化を行った場合が最もスループットが高速となった．汎用コアのみを用いた状況における計測では，Pentium4 上で OpenSSL speed コマンドを実行し，1024byte ずつ暗号化を行った場合が最もスループットが高速となった．これらを比較すると，SPE を 6 基用いた場合の実装コードでは，OpenSSL speed コマンドの約 2.3 倍のスループットであることが確認できた．

ただし，理論値として SPE 6 基のスループットは SPE 1 基の 6 倍の性能であることが考えられるが，計測を行った SPE 6 基のスループットは SPE 1 基の約 5.1 倍であり，理論値と比較して大きく下回っている．これは，SPE の同期をとるために，PPE コード上でのオーバーヘッドが発生するためと考えられる．

また，今回の計測では，AES-CTR モードの暗号処理に要する時間のみを計測したが，実システムで利用するには，暗号の前処理などその他の性能を考慮する必要がある．

5 まとめと今後の課題

本論文では，CBE 上で AES-CTR モードの処理の並列化を試み，そのパフォーマンス計測と評価を行った．今後の課題として，SPE に適したコードへの改良と，実システムでの利用が挙げられる

参考文献

[1] http://cell.scei.co.jp/index_j.html
[2] http://www-128.ibm.com/developerworks/power/cell/docs_documentation.html
[3] http://www.bsc.es/projects/deepcomputing/linuxoncell
[4] http://fp.gladman.plus.com/AES/index.htm
[5] http://www.openssl.org/

[3] OpenSSL0.9.8g の時点では含まれていない．