Vol.2021-HPC-180 No.29
2021/7/21

フルメッシュ接続されたFPGAクラスタによる分散ソートの高速化

賣野 豊[1], 水谷 健二[1], 山口 博史[1], 鯉渕 道紘[2]

概要：これまでに，シリコン・フォトニクスを用いた高帯域密度光トランシーバ「光 IO コア」，およびこのトランシーバと FPGA を用いてノード間をフルメッシュ接続したネットワーク・アーキテクチャ「OPTWEB」等を提案した．今回，8 ノードの FPGA カード間を OPTWEB で接続した FPGA クラスタを用いて，フルメッシュ・ネットワーク越しにノードを跨いだ 64 本のグローバルな並列パイプライン上を key データが 1 回移動するだけでソーティング処理が完結する分散ソーティング・システムを構築し，8 ノードの CPU 間を 100 Gbps の InfiniBand スイッチで接続した CPU クラスタに比べて 57 倍のスループットである 243 Gb/s を実証した．

キーワード：並列計算，ノード間インターコネクト，FPGA アクセラレータ，ハードウエア・ソーティング

Performance enhancement for distributed sorting based on fully connected FPGA cluster

YUTAKA URINO[†1] KENJI MIZUTANI[†1] HIROSHI YAMAGUCHI[†1]
MICHIHIRO KOIBUCHI[†2]

1. はじめに

近年，処理すべきデータ量は指数関数的に増大する一方で，CPU の演算性能の伸びは鈍化しており，そのギャップを埋める手段として，多ノードによる並列分散化およびアクセラレータによるヘテロジニアス化が普及している．このようなシステムでは，アクセラレータ間の通信性能がシステム全体の性能のボトルネックとなり得る．

我々はこれまでこの問題に対して，シリコン・フォトニクスを用いた高帯域密度光トランシーバである「光 I/O コア」およびそれを用いて FPGA ボード間を接続したクラスタ・システムを開発してきた[1]．また，ノード間インターコネクトのトポロジーやルーティングに起因する問題を解決するため，波長ルーティングを用いたインターコネクトである「光ハブ」およびそのフルメッシュ論理トポロジーに適したルーティング・アルゴリズムである「マルチパス・ルーティング」を提案し，並列計算シミュレータを用いて並列計算ベンチマークの実行時間を解析し，アプリケーションの実行時間および並列計算システムの省エネの観点で，光ハブの優位性を示した[2][3]．また，「光ハブ」を行列積計算に適用することで，行列積計算を高速化できることも示した[4]．更に，「光ハブ」をより一般化した「OPTWEB」を提案し，8 ノードの FPGA 間を OPTWEB で接続したクラスタ・システムを構築し，Alltoall や Allreduce などの集合通信や整数ソーティングを高速に実施できることを示した[5][6]．

ソーティングは，リレーショナル・データベース，MapReduce/Spark などのビッグデータ分散処理のためのフレームワーク，光線追跡やレンダリングなどのコンピュータ・グラフィックス，N 体シミュレーションなどの科学技術計算等，その応用先は多岐にわたり，ソーティングのスループットがアプリケーション全体の性能を律速することも多いため，古くから活発に研究されている基本的な演算処理である．

本報告では，文献[6]で報告した整数ソーティングのスループットをベースラインとし，その更なる高速化を論ずる．本報告の構成は以下のとおりである．第 2 章では背景技術として，従来の FPGA 間ネットワークの動向および我々が提案した「OPTWEB」について要点を説明する．第 3 章では，整数ソーティングの手法の一つである分散計数ソートの高速化を理論的に検討する．第 4 章では，第 3 章で理論的に検討した高速化手法を実際に 8 ノードの CPU クラスタおよび FPGA クラスタに実装し，そのスループットを評価し，高速化を検証する．第 5 章では，FPGA のクロック周波数の高速化による更なるスループット向上，ソーティングの key の種類数を増やす方法とその場合のスループット，および他の FPGA ハードウェア・ソータとのスループットの比較について考察する．第 6 章は全体のまとめである．

1 技術研究組合 光電子融合基盤技術研究所
Photonics Electronics Technology Research Association (PETRA)
2 大学共同利用機関法人 情報・システム研究機構 国立情報学研究所
National Institute of Informatics (NII)

ⓒ2021 Information Processing Society of Japan

2. 背景

2.1 従来の FPGA 間ネットワーク

従来の各ノードに FPGA をアクセラレータとして持つ分散システムでは, FPGA 間通信は, FPGA→PCIe 接続→ホスト CPU→ホスト・ネットワーク・インターフェース→外部スイッチを経由したパケット通信（Ethernet または InfiniBand 等）で行うことが一般的であり, その実効帯域幅は PCIe 接続の遅延時間および帯域幅等によって制限されていた[7][8].

近年, 上記の実効帯域のボトルネック解消のため, FPGA 内に Ethernet の知的財産(IP)を実装し，PCIe およびホストを介さずに FPGA 同士が直接パケット・スイッチを介して通信する方法も提案されているが，通信帯域幅の拡大に伴い上記 IP のリソースが肥大化し, 有限な FPGA リソースの制約の下では，更なる広帯域化が困難な状況になっている[9][10][11][12].

更には，外部のパケット・スイッチも排除し，リング，トーラス等のネットワーク・トポロジーを用いた直接網で FPGA 間を接続する提案もされている[13][14]．しかしながら，リング，トーラス等の 1 ノードから直接接続されるノード数（degree 数または radix 数）が少ないネットワークでは，ネットワークの直径および平均最短経路長が長く，マルチ・ホップによる実効帯域幅の低下が懸念される．

2.2 OPTWEB

上記の FPGA 間ネットワークの課題を解決するため，我々は FPGA 間をホストや外部スイッチを介さずに, FPGA 内に実装した軽量な IP を用いて, 広帯域幅でフルメッシュ接続するネットワーク・アーキテクチャ「OPTWEB」を提案した[5]．本システムでは，8 枚の FPGA(Intel Stratix10 MX2100)カードの HBM2 メモリ間で，1 μs 以下の低遅延時間（オーバー・ヘッド）と最大約 800 Gbps/node の広実効帯域幅で，Barrier 同期，Point-to-point 通信および各種の集合通信を実証した．この FPGA カードには，PETRA が開発した 100-Gbps(25-Gbps 送受信×4) ソケット式高密度光トランシーバ「光 I/O コア」が 8 個搭載されている．

2.3 分散計数ソーティング

更に我々は，上記 8 ノードが OPTWEB で接続された FPGA クラスタ・システムに分散計数ソートを実装し，8 台の CPU 間を InfiniBand スイッチで接続した CPU クラスタ・システムに比べて，5.3 倍高速な処理を実証した[6].

3. 分散ソートの高速化

3.1 ソーティング・アルゴリズムの選択

ソーティングには非常に多くのアルゴリズムが知られており，その平均計算量は，例えば比較的ナイーブなバブルソートや挿入ソートで $O(n^2)$, 比較的高速で実用的なクイックソートやマージソートなどで $O(n \log n)$である．ただし n はソートされるキーの個数である．一般に，2 つのキーの間で大小比較を繰り返すアルゴリズムでは，平均計算量を $O(n \log n)$以下にすることは出来ない.

計数ソート(Counting sort)はキー間の比較を行わないアルゴリズムである．計数ソートのアルゴリズムを簡潔に述べると，(1) キーの種類毎の出現数を数え上げ，(2) ソート後にそのキーが格納されるべきメモリ空間上のアドレスを計算し，(3) キーをそのメモリ・アドレスへ移動する，となる．我々の方式は，上記(2)のメモリ・アドレスの計算さえも key 自身の 6 bits の値で代用することで，演算処理を排除し，ほぼデータの移動だけでソーティング処理が完結することが特徴である．複数のノード間で分散計数ソートを行う場合は，上記のメモリ・アドレス空間はノード間を跨ぐグローバル・メモリ空間となり，キーがノード間を移動する際に Alltoallv（送信先毎にメッセージサイズが異なる Alltoall）通信が必要になる．

計数ソートの平均計算量は, キーの種類数を K とすると，$O(n+K)$であり，通常キーの種類数 K はキーの個数 n に比べて小さいことが多いので，その場合計数ソートは他のアルゴリズムに比べて高速である．ただし，キーの種類数 K がメモリ・チャンネル数より大きくなるとメモリへのランダム・アクセスが多発し，キャッシュを持たない FPGA ではメモリ帯域を活かすことが出来なくなる．

そこで我々は，そのような場合には計数ソートを$(\log_k K)$回繰り返す基数ソート(Radix sort)を行うことを想定している．ただし，k は計数ソート 1 回でソートするキーの種類数であり，メモリ・アクセスを効率的に行うためには k は総メモリ･チャンネル数 MN と一致させると良い. ここで，M は 1 ノード当たりの同時に読み込みおよび書き込みが出来るメモリ・チャンネル数，N はノード数である．このとき，基数ソートの平均計算量は，$O((\log_{MN} K)(n+MN))$となる．今回我々がソーティングに用いたシステムでは，1 つの FPGA 当たりのメモリ・チャンネル数が 16 であり，読み込みと書き込みを同時に実施できるメモリ・チャンネル数 M=8，FPGA 数 N=8 であるので，総メモリ・チャンネル数 MN=64 である．ソートの対象を 32-bit 整数，すなわちキーの種類数 $K=2^{32}$ とした場合の，クイックソートまたはマージソートの計算量 $O(n \log n)$と基数ソートの計算量 $O((\log_{MN} K)(n+MN))$の比を図 1 に示す．縦軸は，クイックソートまたはマージソートの計算量を分子，基数ソートの計算量を分母にとっているので，基数ソートがクイックソートまたはマージソートに対して何倍高速かを示している．例えばキー数が 2^{30} の場合，クイックソートやマージソートの計算量は基数ソートの計算量の約 5.6 倍となり，基数ソートの方が約 5.6 倍高速になることが期待され，この比率はキーの数が増えるほど大きくなる．

ⓒ2021 Information Processing Society of Japan

そのため我々は，ソーティングのアルゴリズムとして，上記の計数ソートと基数ソートの組み合わせを選択した．

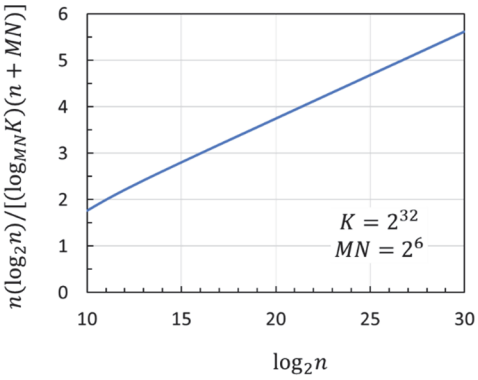


図 1　基数ソートのスループット
（クイックソートまたはマージソートに対する比）

3.2 ベースライン

我々はこれまでに，8 枚の FPGA カード間をフルメッシュ光接続したシステムを用いて分散計数ソートを行い，8 ノードの CPU 間を 100-Gbps の InfiniBand で接続した場合に比べて，約 5.3 倍高速に処理できることを示した[6]．まず，この文献[6]の結果を簡単にレビューする．

文献[6]で報告した FPGA クラスタおよびその比較システムのネットワークの概略を表 1 に示す．我々の試作した FPGA カードは，カード 1 枚当たり最大で 100 GB/s のネットワーク帯域を有するが，この比較実験を行う際には，Bisection 帯域を比較システムとおおよそ合わせるため，ノード間リンク帯域を 4 GB/s，1 ノード当たりの総ネットワーク帯域（以下，ノード帯域と呼ぶ）を 32 GB/s に絞って実験を行った．

表 1　FPGA／CPU クラスタのネットワークの比較[6]

	FPGAクラスタ	CPUクラスタ（比較システム）
ノード数	8	8
ノード間トポロジー	フルメッシュ（OPTWEB）	ツリー（InfiniBand）
ノード間リンク帯域（GB/s）	4	12.5
ノード帯域（GB/s）	32	12.5
Bisection帯域（GB/s）	64	50
プロセッサ	FPGA（Stratix10 MX2100）	CPU（Xeon Gold 5215）

ソーティング処理されるデータは，列型データベースを意識し，key と value のセットとし，それぞれ 4-byte 整数とした．ただし，key の種類数は 64（key の値は 0～63）である．ソーティング処理では，key と value のセット（レコード）が key の昇順に並び替えられる．FPGA クラスタの場合，ソーティング処理後には，例えば key が 0 のレコードはノード 0 のメモリ・チャンネル 0 に，key が 63 のレコードはノード 7 のメモリ・チャンネル 7 に格納される．

FPGA に実装した分散計数ソートの工程フローを図 2 に示す．この工程フロー図では，key または value データのメモリへの read/write 単位でフローを分割している．各ステップで，データの read と write が対になっている場合は，read のメモリ・チャンネルおよびメモリ領域と write のメモリ・チャンネルおよびメモリ領域が重ならないようにしており，read から write までは 1 本のパイプラインとして同時に実施することが出来る．そのためメモリ・アクセス回数（パイプラインのシリアル本数）は，read と write の対で 1 回と数えると，この図 2 の工程数と同じ 11 となる．以下に図 2 の各工程を簡単に説明する．

工程①では，key データをメモリから read し，key の種類ごとに出現回数を数え上げ，key の種類ごとの出現度数表（ヒストグラム）を作成し，フリップ・フロップに保存する．このヒストグラムは後の工程⑥と⑦で，Alltoallv 通信を行う際に，送信先毎の key 数として利用される．

工程②では，key データを read し，その上位 3 bits から，次工程③でその key が移動する行先メモリのメモリ・チャンネル番号(0～7)の表を作成しブロック RAM に保存する．

工程③では，key を read し，工程②で作成した表に従ってクロスバー・スイッチを切り替え，key を行先メモリに write する．なお，工程②と③は，ブロック RAM の容量を節約するため，512-burst 毎にブロック RAM を上書きしながら行う．

工程④は工程②と同じ動作であり，工程⑤は工程③の key を value に置き換えた動作になる．

工程⑥では，送信元ノードで key を read し，ノード間で Alltoallv 通信を行い，送信先ノードで key を write する．この際，工程①で作成したヒストグラムを用いて，各送信先に何個の key を送るかの情報を獲得し，Alltoallv 通信の終了判定に利用している．ノード間はフルメッシュ接続されており，各ノード間は専用のリンク帯域が確保されている．また，予め工程③で送信先ノードに対応したメモリ・チャンネルに接続されたメモリ領域に key を移動済みであるため，メモリからネットワークポートまでも専用のバスが確保されている．これらの仕組みにより，送信元メモリから受信先メモリまで専用の帯域を使ったデータの移動が可能になる．

工程⑦では，工程⑥の key を value に置き換えた動作を行う．

工程⑧⑨および工程⑩⑪では，工程②③および工程④⑤の上位 3 bits を下位 3 bits に置き換えた動作をそれぞれ行う．このような仕組みにより，工程②③，工程④⑤，工程⑧⑨および工程⑩⑪に，ほぼ同じ回路を用いることができ，FPGA の回路規模の節約になる．

ⓒ2021 Information Processing Society of Japan

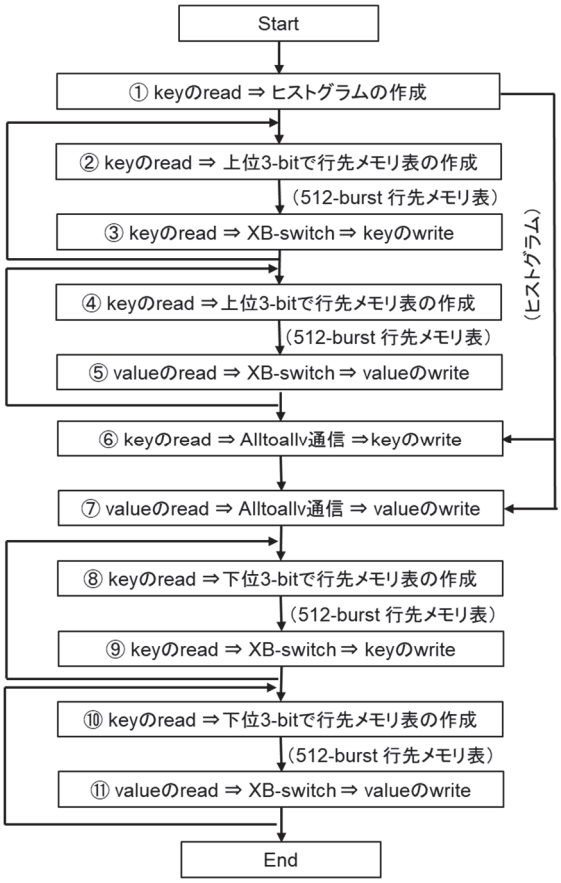


図 2　文献[6]の分散計数ソートの工程フロー

実験では，key と value の対（レコード）の数は 2^{27} 個，key の種類数は 64（key の値は 0～63），key の分布はランダムとした．このとき，FPGA クラスタで得られたソーティングのスループット（key の総 byte 数をソーティング処理時間で割った値）は，17.3 GB/s であった．このスループットを本報告におけるベースラインとし，以下にその高速化について述べる．

3.3 スループットの定式化

我々の方式の分散ソーティングでは演算時間はほぼ無視できるため，分散ソーティングのスループットは，主にメモリ・アクセスとノード間通信の遅い方に律速され，その理論ピーク性能 T は，

$$T = N \min\left(\frac{WMC}{A}, \frac{B}{m}\right) \quad (1)$$

で表される．ここで，N はノード数，W は 1 メモリ・チャンネル当たりのバス幅，M は 1 ノード当たりのメモリ・チャンネル数， C はクロック周波数，A はメモリ・アクセス回数，B はノード帯域，m は通信回数であり，min 関数内の第 1 項および第 2 項はそれぞれメモリ・アクセスおよびノード間通信に起因するスループットである．メモリ・アクセス律速の場合は，一度メモリから読み出したデータをなるべく長いパイプラインで処理した後にメモリに書き込むことにより，メモリ・アクセス回数を削減することが高速化に有効である．

文献[6]では, 式(1)の各パラメータの値は, N=8, W=32 byte, M=8, C=125 MHz, A=11, B=32 GB/s, m=2 であり，この場合はメモリ・アクセス律速となり，期待される理論ピーク性能は 23 GB/s，実測値 17.3 GB/s は理論ピーク値の 74%だった．

仮に図 2 の全ての処理工程を 1 本のパイプラインに併合し，メモリ・アクセス回数およびノード間通信回数をそれぞれ 1 回(A=m=1)まで削減できた場合，式(1)の min 関数内の第 1 項および第 2 項はいずれも 32 GB/s で同じになる．したがって，クロック周波数の高速化などによるメモリ帯域の更なる拡大がある場合は，メモリ帯域拡大と並行してネットワーク帯域の拡大も必要となる．

3.4 高速化手法の提案

本節では，前記のベースラインに対して，以下の通りネットワーク帯域の拡大，value 処理の削除，クロック周波数の変更，メモリ・アクセス削減を行うことで，OPTWEB における分散計数ソートの高速化を検討する．

(1) ネットワーク帯域の拡大

前述の通り，文献[6]では比較対象となるシステムとのフェアな比較のため，ノード帯域を 32 GB/s に絞って実験したが，前節の議論を踏まえ，本報告ではこのノード帯域を可能な限り拡大することにした．

ノード帯域の上限は，主に FPGA のロジック・ユニット（Intel 製 FPGA の場合，Adaptive Logic Module, ALM と呼ばれる）の量によって制限される．図 3 に FPGA の ALM の使用率を示す．左および中央の棒は，FPGA にアプリケーション･ロジックが実装されていない状態, つまり FPGA カードをネットワーク・インターフェース・カード(NIC)として使う場合の ALM の使用率であり，左はノード帯域が 50 GB/s，中央は 100 GB/s の場合である．ノード帯域が 100 GB/s の場合，ALM の使用率は約 60%となり，アプリケーション･ロジックを実装するスペースが十分とは言えない．そこで今回は，ノード帯域を 50 GB/s とした．この場合，仮にメモリ・アクセスおよびノード間通信が共に 1 回になったとしても，式(1)の min 関数内の第 1 項は 32 GB/s，第 2 項は 50 GB/s となり，ネットワーク帯域律速になることはない．

図 3 の右の棒は，ノード帯域を 50 GB/s とし，分散計数ソートのロジック・コアを実装した場合の ALM 使用率を示しており，全体で約 70%程度となった．ALM 使用率とクロック周波数の間にはトレードオフの関係があり, ALM 使用率を上げ過ぎると，クロック周波数を維持することが困難になるため，この程度の ALM 使用率が適当であると考えている．

ⓒ2021 Information Processing Society of Japan

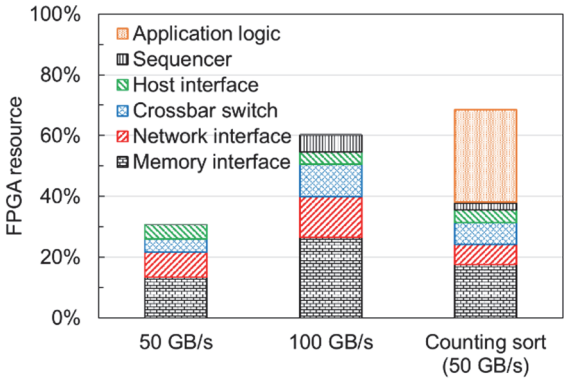


図 3　FPGA の ALM の使用率

(2) value 処理の削除

一般にデータベースでは 1 つの key に対して複数の value が付随してレコードを形成することが多いが，ソーティングの処理能力を議論する場合には，value の処理を含めると話が複雑になるため，key のみの処理で議論することも多い．そこで本報告でも，以後は value の処理を削除し，key のみを扱うこととした．value の処理を削除すると，図 2 に示した 11 の工程の内，④⑤⑦⑩⑪の 5 つの工程を削除することができ，図 4 に示す通りメモリ・アクセス回数も 11 回から 6 回に削減することができる．

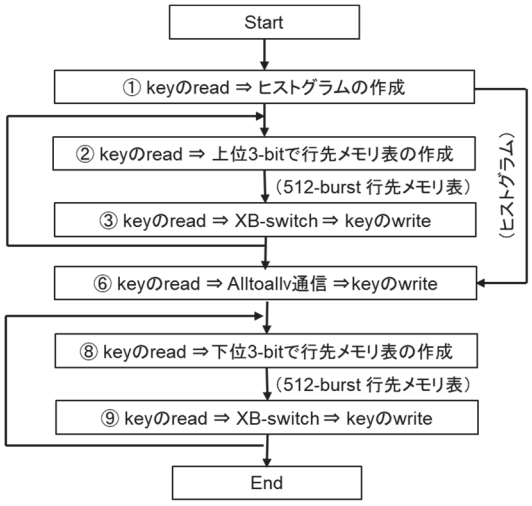


図 4　value 処理削除後の分散計数ソートの工程フロー

(3) クロック周波数の変更

value 処理の削除などの回路規模の削減により，FPGA のクロック周波数を従来の 125MHz から 150 MHz に上昇させることが出来た．ただし，クロック周波数をどこまで高くできるかは，FPGA 内の回路の規模や配置に強く依存するため，今後の高速化検討の中で常にクロック周波数が 150 MHz で一定ではないことは注意する必要がある．

上記の value の削除，ネットワーク帯域の拡大およびクロック周波数の 150 MHz 化を実施した時点で，式(1)の各パラメータは N=8, W=32 byte, M=8, C=150 MHz, A=6, B=50 GB/s, m=2 であり，これにより期待される理論ピーク性能は 51 GB/s となる．

(4) 行先ノード／メモリ表作成工程のスイッチ工程への併合によるメモリ・アクセス削減

図 4 に示す Value 削除後の工程①②③⑥⑧⑨の 6 工程内，工程①のヒストグラム（行先ノード vs key 数の表）作成工程および工程②の行先メモリ表の作成工程を，工程③のクロスバー・スイッチ切り替え工程に併合して 1 本のパイプライン化した．また，工程⑧の行先メモリ表の作成工程を，工程⑨のクロスバー・スイッチ切り替え工程に併合して 1 本のパイプライン化した．これらにより，図 5 に示す通り，メモリ・アクセス回数は 3 回に削減され，スループットの理論ピーク性能は 102 GB/s となる．

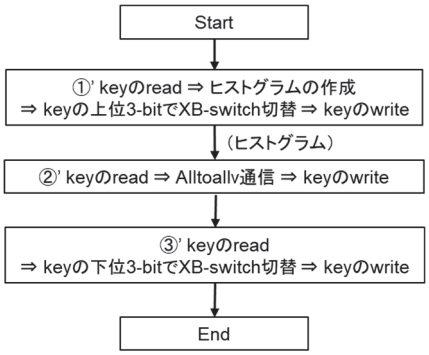


図 5　行先ノード／メモリ表作成工程併合後の分散計数ソートの工程フロー

(5) 通信終了検知方式変更によるメモリ・アクセス削減

従来，実際に送信または受信されたメッセージサイズをカウントし，シーケンサで指定のメッセージサイズと比較することで通信の終了を検知していたが，その代わりに EOP(End of Packet)を送受信することで通信終了を検知する方式に変更した．これにより，図 5 の②'工程の Alltoallv 通信の際に事前に送信先毎のメッセージサイズを確定する必要がなくなり，図 5 の工程①'のヒストグラムの作成が不要となった．そこで①'と②'を 1 本のパイプラインに併合し，図 6 に示すとおりメモリ・アクセス回数を 2 回に削減した．これにより，スループットの理論ピーク性能は 154 GB/s となる．

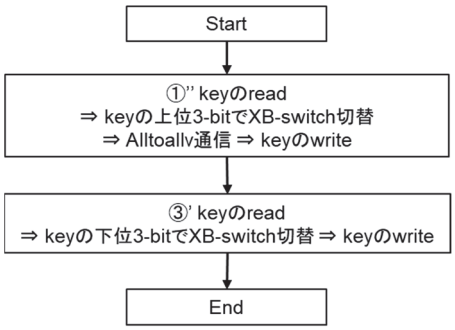


図 6　通信終了検知方法変更後の分散計数ソートの工程フロー

ⓒ2021 Information Processing Society of Japan

(6) 書込み領域振り分けによるメモリ・アクセス削減

従来，他のノードから受信した key データは，その受信元ノードに対応した専用のメモリ領域に書き込まれていた．この方式を，他のノードから受信した key データの下位 3 bits の値に従って 8 つのメモリ領域の内の対応するメモリ領域に振り分けて書き込む方式に変更した．また，図 6 の①”の XB-switch 回路と③’の XB-switch 回路は同じ回路を使用していたため，工程①”と工程③’を同時に実行することは出来なかった．そこでこの XB-switch 回路を各 FPGA 内に 2 つ用意し，工程①”と工程③’を同時に実行出来るようにした．これにより，図 7 に示すように，全ての工程が 1 本のパイプラインに併合され，メモリ・アクセス回数は 1 回に削減される．これは，フルメッシュ・ネットワーク越しにノードを跨いだ 64 本のグローバルな並列パイプライン上を key データが 1 回移動するだけでソーティング処理が完結することを意味する．このとき，スループットの理論ピーク性能は 307 GB/s となる．

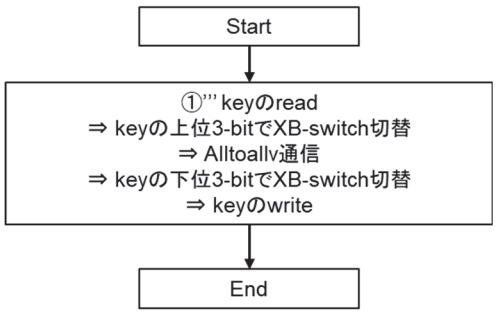
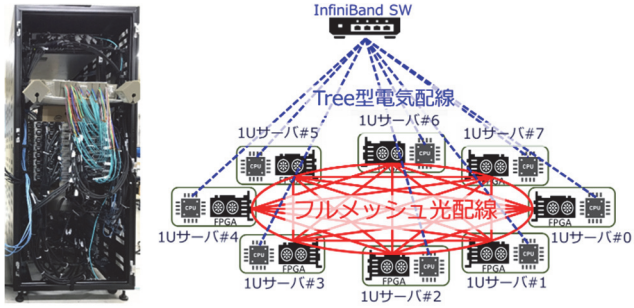


図 7　書込み領域振り分け後の
分散計数ソートの工程フロー

4. 性能評価

4.1 実験系の構成

フルメッシュ光配線(OPTWEB)で接続された 8 枚の FPGA(Intel Stratix10 MX2100)カードで構成された FPGA クラスタに，3 章で示した各高速化手法を順次実装し，分散計数ソートのスループットを評価した．FPGA 回路の合成には，Intel Quartus Prime Pro version 19.4 を用いた．比較のため，8 ノードの CPU(Intel Xeon Gold 5215, 2.5 GHz)間を 100 Gbps の帯域で接続した CPU クラスタ・システムでも同様の評価を行った．各 CPU ノードは 8 個の 187.2-Gbps 帯域を有する DDR4 メモリを持ち，1 台の InfiniBand (IB) スイッチ(Mellanox SB7700)を経由して相互接続している．CPU クラスタの OS は CentOS Linux v7.9.2009，C コンパイラは gcc 6.3.0，MPI は Open MPI v4.1.0 を用いた．構築した 8 ノードのラック・サーバ・システムの写真を図 8(a)に，構成の模式図を図 8(b)に示す．また，FPGA クラスタおよびその比較システムである CPU クラスタのネットワークの概略を表 2 に示す．Key の数は 2^{27} 個，key の種類数は 64（key の値は 0～63），key の分布はランダムとした．



(a)　　(b)

図 8　分散計数ソート評価用ラック・サーバ・システム

表 2　FPGA／CPU クラスタのネットワークの比較

	FPGAクラスタ	CPUクラスタ（比較システム）
ノード数	8	8
ノード間トポロジー	フルメッシュ（OPTWEB）	ツリー（InfiniBand）
ノード間リンク帯域 (GB/s)	6.25	12.5
ノード帯域 (GB/s)	50	12.5
Bisection帯域 (GB/s)	100	50
プロセッサ	FPGA（Stratix10 MX2100）	CPU（Xeon Gold 5215）

4.2 CPU+IB でのスループット評価

CPU 間を 100 Gbps の InfiniBand Switch で Tree 型に接続したシステムにおいて，1 ノード当たりのコア数およびノード数を変化させたときの分散計数ソートのスループット（実測値）を図 9 に示す．この実験では，1 ノード当たりの key 数を 2^{24} で一定としているため，コア数に対しては強スケーリング，ノード数に対しては弱スケーリングとなっている．1 ノード当たりのコア数が 1 の場合，ソーティング処理は計算律速となるため，スループットはノード数にほぼ比例してスケールする．1 ノード当たりのコア数が 4～8 の場合，ノード数の増加に従ってスループットも増加するが，ネットワーク帯域の制限のため，増加率に飽和傾向が見られる．1 ノード当たりのコア数が 16～20 の場合，詳しい理由は不明であるが，ノード数の増加に伴いスループットが低下する現象が見られた．そこで今回は，これらの中で最高のスループットが得られた，1 ノード当たりのコア数が 8 で，8 ノードの CPU クラスタを比較システムとして用いた．

ⓒ2021 Information Processing Society of Japan

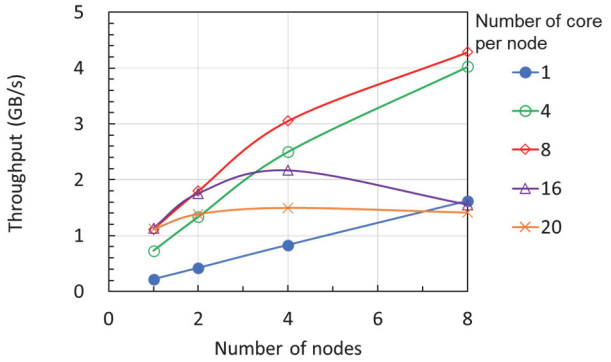
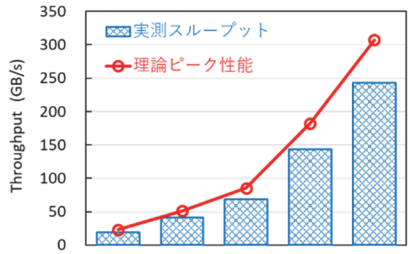


図 9　CPU クラスタの分散計数ソートのスループット（実測値）

4.3 FPGA+OPTWEB のスループットの評価

3 章で述べた各高速化手法における FPGA クラスタを用いた分散計数ソートのスループットを図 10 に示す．メモリ・アクセス回数が 11→6→3→2→1 と減少するに伴い，クロック周波数は, 125→150→125→178→150 MHz と変化し，その結果式(1)で求まる理論ピーク性能は，23→51→85→182→307 GB/s となった．全ての場合で，メモリ帯域律速である．これらに対して，実測のスループットは，19→41→69→144→243 GB/s となり，概ね理論ピーク性能の 80%の結果となった．



メモリ・バス幅 W (Byte)	32	32	32	32	32
メモリ・チャンネル数 M	8	8	8	8	8
ノード数 N	8	8	8	8	8
クロック周波数 C (MHz)	125	150	125	178	150
メモリ・アクセス回数 A	11	6	3	2	1
理論ピーク性能 (GB/s)	**23**	**51**	**85**	**182**	**307**
実測スループット (GB/s)	**19**	**41**	**69**	**144**	**243**
対理論ピーク比	80%	80%	80%	79%	79%

図 10　FPGA クラスタの分散計数ソートのスループット

次に，FPGA クラスタ（メモリ・アクセス数は 1）と CPU クラスタ（8 コア／ノード）のスループットの比較を図 11 に示す．前述の通り，CPU クラスタはネットワーク帯域制限のため，8 ノードでのスループットは 1 ノードの場合の 3.8 倍しか得られず，ノード数に対してスケールしていない．8 ノード同士で CPU クラスタと FPGA クラスタを比較すると，FPGA クラスタの方が 57 倍高速となった．これらの結果は，分散システムにおけるネットワーク性能の重要性を示唆している．また，仮に CPU クラスタのネットワーク実効帯域幅が十分広く，8 ノードでのスループットが 1 ノードの場合の 8 倍になると仮定した場合でも（図 11 の右から 2 番目の棒），FPGA クラスタは CPU クラスタに対して 27 倍高速である．CPU のクロック周波数(2.5 GHz)が FPGA のそれ(150 MHz)に比べて約 17 倍高速であるにもかかわらず，この結果となったことは，ノイマン型コンピューティングに対するデータフロー型コンピューティングの高速性およびヘテロジニアス・コンピューティングの有効性を示している．

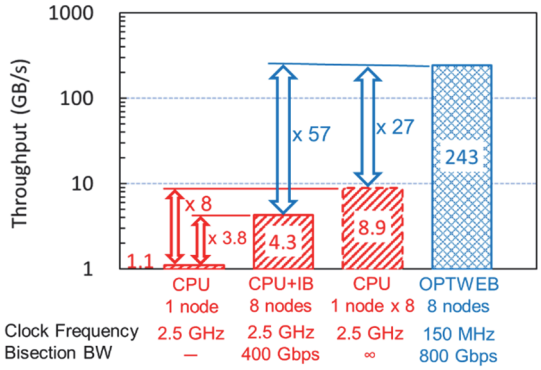


図 11　CPU／FPGA クラスタの分散計数ソートのスループット比較

5. 考察

5.1 クロックの高周波数化による高速化

今後の FPGA のクロックの高周波数化による分散計数ソートの高速化の余地を検討する．今回実測で得られた分散計数ソートのスループットの最高値は 243 GB/s，このときのクロック周波数 150 MHz から計算される理論ピーク性能は 307 GB/s，実測値は理論ピーク値の 79%だった．メモリ・アクセスおよびノード間通信の回数がいずれも 1 回の場合，分散ソートのスループットはメモリ帯域とネットワーク帯域の狭い方によって制限される．現在の（ネットワークの）ノード帯域は 50 GB/s であるから，(1 ノード当たりの)メモリ帯域が 50 GB/s まではメモリ帯域律速である．メモリ帯域が 50 GB/s となるクロック周波数を逆算すると，195 MHz となる．したがって，クロック周波数が 195 MHz までは，クロック周波数に比例した分散ソートの高速化が実現出来ると見込まれる．クロック周波数が 195 MHz のときも実測スループットが理論ピーク値の 79%になると仮定すると，実測スループットは 316 GB/s になると予測される．

5.2 基数ソートによる key 種類数の拡大

これまでのソーティングの実験は，key の種類を 0～63 の 64 種類で行った．これは 8 ノードの FPGA クラスタ全体で同時に read/write 出来るメモリ・チャンネル数が 64 であることに起因している．ソーティングの応用例の中には，key の種類数が 64 あれば十分な応用例もあることが期待され，そのような応用例では上記の通り高速なスループット

ⓒ2021 Information Processing Society of Japan

を実現できる．一方，より多数の種類の key をソーティングする必要がある場合の対処方法としては，ハードウェア的対処法とソフトウェア的対処法がある．

ハードウェア的対処法として，FPGA 1 個当たりのメモリ帯域を増やす scale-up 的方法と FPGA 数を増やす scale-out 的手法がある．前者に関しては，今後も FPGA ベアチップ内の集積度の向上や 3 次元実装などによるパッケージ・モジュールの高密度化などが期待出来るが，FPGA ベンダーの動向に依存する．後者に関しては，我々が提案しているフルメッシュ光接続(OPTWEB)の場合，最も単純な 1 階層または 1 次元の構成でも最大 64 ノード程度まで scale-out が可能であり，更に多階層または多次元化することで，より大規模な scale-out が可能であると予測している[3][4][5]．これらの方法は，いずれも空間的並列度を拡大させる方法であり，key 種類数の拡大と同じ比率で空間的並列度も拡大させる限り，スループットの低下は伴わない．

現状のハードウェアを変更せずにソフトウェア的に対処する方法としては，基数ソートの導入がある．前述の通り，1 回の計数ソートでソーティングする key の種類数が 64 (=2^6)で，基数ソートする key の種類数が K の場合，基数ソートに必要な計数ソートの回数は $\log_2 K/6$ 回となり，この回数に反比例して基数ソートのスループットは低下する．つまり，処理時間の K 依存性は $O(\log K)$である．FPGA クラスタの key の種類が 64 の時の計数ソートのスループットの実測値(243 GB/s)から，key 種類数を変えた場合の基数ソートのスループットを予測した結果を図 12（青破線）に示す．

一方，CPU クラスタを用いた計数ソートの処理時間の key 種類数 K に対する依存性は，$O(n+K)$であり，K の増加によるスループットの低下は，CPU クラスタを用いた計数ソートの方が大きくなる．実際に，CPU クラスタを用いて key 種類数 K を変えた場合の計数ソートのスループットを測定した結果を図 12（赤●）に示す．なお，K=2^{25} 以上では，プログラムの処理中にエラーが発生したため，K=2^{24} までをプロットした． Key の種類数に依らず，常に FPGA クラスタを用いた基数ソートの方が，CPU クラスタを用いた計数ソートより高速になる．CPU の計数ソートで最も高速となったのは，key の種類数が 128(=2^8)のときなので，CPU でこの計数ソートを繰り返す基数ソートを行った場合の予測値も図 12（赤一点鎖線）で示す．この場合計数ソートのような key 種類数の増加に伴う急激なスループットの劣化を見られないが，FPGA 基数ソートに比べて低速であることは変らない．

計数ソートを複数回繰り返す基数ソートには，key の最上位桁(Most Significant Digit)から順に計数ソートを行う MSD 基数ソートと，key の最下位桁(Least Significant Digit)から順に計数ソートを行う LSD 基数ソートがある．MSD 基数ソートは，繰り返し回数に従ってソートの範囲が狭くなるため，key データの移動の局所性が高まり，データ移動時間を短縮出来る可能性があるが，各繰り返しを同一の回路で実施することが難しく，回路リソースが限られている FPGA への実装に向いていないと考えている．一方 LSD 基数ソートは，ソートの対象となる桁が変わること以外は，各繰り返しはほぼ同じ動作であり，全ての繰り返しを同一の回路で実施可能である点が FPGA への実装に適している．

ただし，LSD 基数ソートでは，各計数ソートが安定ソートである必要がある．現時点で，我々が FPGA クラスタに実装した計数ソートは安定ソートになっていない．そこで現在我々は，計数ソートの安定ソート化を行い，その計数ソートを繰り返すことで LSD 基数ソートを実現するための開発を行っている．

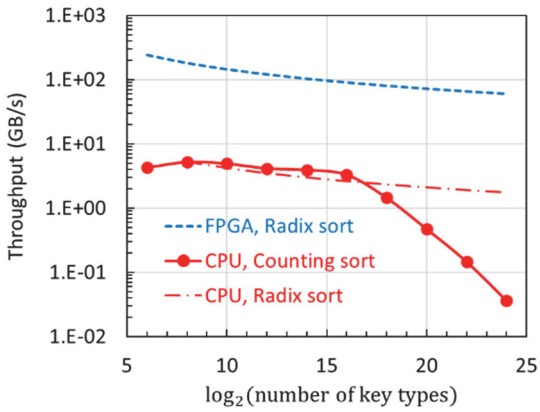


図 12　分散計数ソートおよび分散基数ソートのスループットの key 種類数依存性

5.3 他の FPGA ハードウェア・ソータとの比較

他の FPGA を用いたハードウェア・ソータとのスループットの比較として，文献[15]で報告されている Bonsai と比較を行った．FPGA ハードウェア・ソータの性能は FPGA の回路規模に依存し，その進歩は速いため，出来る限り最新の報告として文献[15]を選択した．比較結果を図 13 に示す．文献[15]の TABLE I に依ると，Bonsai によるソーティング処理時間は，1 GB 当たり 172 ms であり，この値からスループットを計算すると，5.8 GB/s となる．なお，上記 TABLE I には他に 6 つのソータが引用されているが，その中で Bonsai が最も高速である．上記の値は 1 ノードでの値なので，仮に 8 つの FPGA を十分広い帯域幅を持つネットワークを使って接続し，1 ノードのスループットの 8 倍のスループットが得られたと仮定した場合，そのスループットは 46 GB/s となる．

Bonsai の key の範囲は 32 bits なので，それと比較するために前節で示したように，我々の FPGA クラスタで計数ソートを 6 回繰り返す基数ソートを行った仮定すると，スループットは今回の実験結果の 1/6 の 40 GB/s になると予測される．したがって，我々の FPGA クラスタは，Bonsai の

ⓒ2021 Information Processing Society of Japan

1 ノードでのスループットの 8 倍のスループットとほぼ同程度であることが分かる．仮に 5.1 節で検討したように，我々の FPGA のクロック周波数を 195 MHz まで高速化できた場合，基数ソートのスループットはクロック周波数に比例して 53 G/s となり，Bonsai を上回ると予測される．ちなみに，Bonsai のクロック周波数は 250 MHz である．

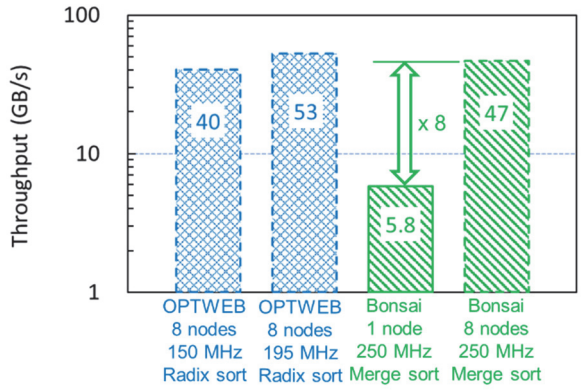


図 13　OPTWEB と Bonsai のスループットの比較

6. まとめ

これまでに，分散システムにおけるノード間通信ボトルネックを解消するため，シリコン・フォトニクスを用いた高帯域密度光トランシーバ「光 IO コア」，波長ルーティングを用いたノード間インターコネクト「光ハブ」，ノード間をフルメッシュ接続したネットワーク・アーキテクチャ「OPTWEB」，OPTWEB を用いた FPGA クラスタ・システム等を提案した．

ソーティングは応用範囲の広い計算であり，その高速化が期待される．そこで今回，上記 FPGA クラスタ・システムを用いた分散ソーティングの高速化を行った．

具体的には，8 ノードの FPGA 間を OPTWEB で接続した FPGA クラスタを用いて，フルメッシュ・ネットワーク越しにノードを跨いだ 64 本のグローバルな並列パイプライン上を key データが 1 回移動するだけでソーティング処理が完結する分散ソーティング・システムを構築し，8 ノードの CPU 間を 100 Gbps の InfiniBand スイッチで接続した CPU クラスタに比べて 57 倍のスループットである 243 Gb/s を実証した．

ただし，この実験では key の種類が 6 bits であるため，32 bits の整数をソーティングするためには，この計数ソートを 6 回繰り返す基数ソートを実行する必要があり，その場合のスループットは 1/6 の 40 GB/s になると予測される．このスループットは文献[15]の Bonsai（1 ノード）の 8 倍のスループットとほぼ同程度であり，今後のクロック周波数の高周波数化により，これを上回るスループットも実現可能だと考えている．

謝辞　この成果は，国立研究開発法人新エネルギー・産業技術総合開発機構(NEDO)の委託業務(JPNP13004)の結果得られたものです．FPGA 間ネットワーク(OPTWEB)および分散計数ソートの FPGA への実装にご尽力頂いた，NEC プラットフォームズ株式会社の藤原博志氏，秋吉賢治氏，日本システムウエア株式会社の石田好延氏，阪本貴仁氏に感謝致します．

参考文献

[1] Nakamura, T., Yashiki, K., Mizutani, K., Nedachi, T., et al.. Fingertip-Size Optical Module, “Optical I/O Core”, and Its Application in FPGA. 2019, IEICE Trans. Electron., vol. E102–C, no.4, p.333–339.
[2] 賣野豊，水谷健二，臼杵達哉，中村滋．ノード間波長ルーティング・インターコネクト（光ハブ）を用いた並列計システム．信学技報, 2019, CPSY2019-33, DC2019-33.
[3] Urino, Y., Mizutani, K., Usuki, T., and Nakamura, S.. Wavelength-routing interconnect "Optical Hub" for parallel computing systems. 2020, HPC2020, p.81–91.
[4] 賣野豊．ノード間光ハブ接続を用いた並列行列積計算の高速化．情処学会研究報告, 2020, Vol.2020-HPC-175, No.11.
[5] Mizutani, K., Yamaguchi, H, Urino, Y., and Koibuchi, M.. OPTWEB: A Lightweight Fully Connected Inter-FPGA Network for Efficient Collectives. IEEE Transactions on Computers, Vol.70, No.6, p.849–862, June 2021.
[6] Mizutani, K., Yamaguchi, H, Urino, Y., and Koibuchi, M.. Accelerating Parallel Sort on Tightly-Coupled FPGAs enabled by Onboard Si-Photonics Transceivers. Optical Fiber Communication Conference. 2021, Th5H.1.
[7] Duato, J., Yalamanchili, S., and Ni, L.. Interconnection Networks: An Engineering Approach, Amsterdam, The Netherlands: Morgan Kaufmann, 2002.
[8] Azegami, K., Musha, K., Hironaka, K., Ahmed, A., et al.. A STDM (static time division multiplexing) switch on a multi FPGA system. in Proc. IEEE 13th Int. Symp. Embedded Multicore/Many-Core Syst.–Chip, 2019, pp. 328–333.
[9] Fang, J., Mulder, Y., Hidders, J., Lee, J., et al.. In-memory database acceleration on FPGAs: A survey. VLDB J, vol. 29, no. 1, pp. 33–59, 2020.
[10] Lant, J., Navaridas, J., Jujan, M., and Goodacre, J.. Toward PGA based HPC: Advancing interconnect technologies. IEEE Micro, vol. 40, no. 1, pp. 25–34, Jan./Feb. 2020.
[11] George, A., Herbordt, M., Lam, H., Lawande, A. et al.. Novo-G#: Large-scale reconfigurable computing with direct and programmable interconnects. in Proc. IEEE High Perform. Extreme Comput. Conf., 2016, pp. 1–7.
[12] Matteis, T., Licht, J., Beranek, J., and Hoefler, T.. Streaming message interface: High-Performance distributed memory programming on reconfigurable hardware. in Proc. Int. Conf. High Perform. Comput, Netw. Storage Anal., 2019, pp. 1–33.
[13] Mondigo, A., Ueno, T., Sano, K., and Takizawa, H.. Scalability analysis of deeply pipelined tsunami simulation with multiple FPGAs. IEICE Trans. Inf. Syst., vol. E102-D, no. 5, pp. 1029–1036, 2019.
[14] Stern, J., Xiong, Q., Skjellum, A., and Herbordt, C.. A novel approach to supporting communicators for in-switch processing of MPI collectives. in Proc. Workshop Exascale MPI, 2019, pp. 1–10.
[15] Samardzic, N., Qiao, W., Aggarwal, V., Chang, M., et al.. Bonsai: High-Performance Adaptive Merge Tree Sorting. 2020 ACM/IEEE 47th Ann. Intern. Symp. on Computer Architecture, pp. 282–294.

ⓒ2021 Information Processing Society of Japan