研究論文 Received February 2, 2000; Accepted March 25, 2000

光学 29, 5 (2000) 316-322

フレネル型計算機ホログラムの高速計算のための分散並列処理

由井薗隆也・西　省吾・森　邦彦・中山　茂

鹿児島大学工学部情報工学科　〒890-0065 鹿児島市郡元 1-21-40

Distributed Parallel Processing for High Speed Calculation in Computer-Generated Fresnel Hologram

Takaya YUIZONO, Syogo NISHI, Kunihiko MORI and Shigeru NAKAYAMA

Department of Information and Computer Science, Faculty of Engineering, Kagoshima University, 1-21-40, Koorimoto, Kagoshima 890-0065

A computer-generated Fresnel hologram is one of computer hologram and allows us to reconstruct three-dimensional distribution of light wave. We construct parallel processing with network of personal computers to get high-speed calculation of the Fresnel hologram. Software to perform the process was JavaSpaces™ by Java language that supports generative communication on object shared spaces similar to tuplespace. The result of the parallel processing shows that the number of task partition affects its perfo rmance. Adaptive number of task partition provides good load balancing and adaptive performance of the computer network system. A lot of the partitions provides communication overload on a JavaSpaces server and diminishes the system performance hardly. Furthermore, we add a fault tolerant function to reduce an influence of very slow computer and to remove worker's trouble.

1. はじめに

計算機ホログラムは計算機内で表現できる任意の波面を再生できるため，融通性が高い，再現性が高いなどの従来のアナログホログラムにはない特徴を有しており，各種の研究がこれまでになされている[1-4]．特にフレネル型計算機ホログラムは光波の3次元分布を自由に再生できる．フレネル型計算機ホログラムは物体からのフレネル回折波を計算し，参照光との干渉計算を行うことによって作成できる．しかしながら，フレネル回折式は3重積分で定義され，直接計算するには膨大な演算時間を要し，現実的にフレネル回折式から3次元物体の計算機ホログラムを作成することは困難であった．これまでに3次元物体を奥行き方向にサンプリングし，サンプリングされた複数の2次元平面を再生することによって3次元物体を再生する方法が提案されている[4]．この方法では高速フーリエ変換アルゴリズムが使用できるため，奥行き方向のサンプリング数が少ない場合高速に作成することが可能である．しかしながら，奥行き方向のサンプリング数が増加すればそれに伴い計算量も増加していき，現実的にサンプリング数は制限されたものにならざるを得ず，連続的に奥行き情報が変化する3次元物体の再生は困難であった．

3次元物体が定義されたとき，ホログラム面における各点のフレネル回折波は独立に計算することができ，並列演算およびネットワークを用いた分散処理が本質的に可能である．これまでに計算機ネットワークによるフレネル型計算機ホログラムの並列演算手法が報告されており[5]，分散並列処理による高速演算の可能性が示唆されている．しかし，特殊なTCPプロトコルを使用し，同じ種類のコンピューターを特定して均一なジョブを与えて行われていた．

そこで，CPUやOSの異なった種類のコンピューターを組み合わせた大規模な分散並列処理を提供しうる計算機ネットワークを用いた実現に注目した．その処理を行うために，オブジェクト指向型Java言語におけるLindaモデル[6]の実装であるJavaSpaces[7]を用いた．分散並列処理を行うためには，ワーカー複製形式という同一プログラムによる処理を大量データに対して加える並列処理技法を用いた．Lindaはタップル空間と呼ばれる抽象空間を用いてお互いの情報伝達を行うので，特定の相手を明示して通信接続する必要はなく，柔軟な分散並列処理の実行に適している．

E-mail: yuizono@ics.kagoshima-u.ac.jp

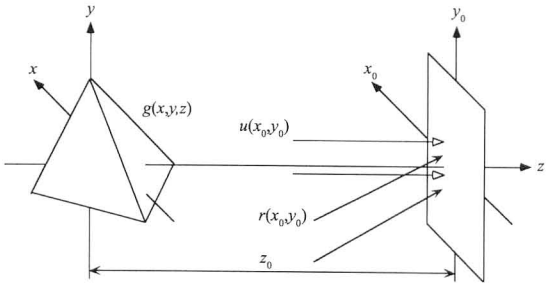

Fig. 1. Recording model of Fresnel hologram.

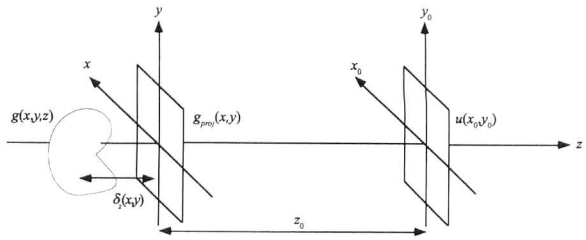

Fig. 2. Computer-generated Fresnel hologram using projected image.

本論文では，柔軟な分散並列処理の構築を提供しうるLindaモデルのJavaへの実装であるJavaSpacesを用いて，フレネル回折波の演算を行う方法について提案し，実験を行った．JavaSpacesを利用することにより，拡張性と汎用性を有したフレネル型計算機ホログラムの分散並列処理が可能になる．

2. フレネル型計算機ホログラム

Fig. 1のようにホログラム面の座標系を (x_0,y_0) とし，3次元物体 $g(x,y,z)$ を記録するフレネル型ホログラムを考える．z 軸は (x_0,y_0) 平面を基準とした光軸方向にとる．このとき，物体の (x,y) 上の広がりが z に比較して十分小さい，いわゆる3次元物体 $g(x,y,z)$ のフレネル回折領域にホログラム面があるとすると，ホログラム面での物体波 $u(x_0,y_0)$ すなわちフレネル回折波は

$$u(x_0,y_0)=\iiint g(x,y,z)\exp\left[-\frac{ik}{2z}\{(x-x_0)^2+(y-y_0)^2\}\right]\mathrm{d}x\mathrm{d}y\mathrm{d}z \quad (1)$$

と表現でき，ここで $k=2\pi/\lambda$ とする．これは z 軸に関する積分があり，このまま3次元物体のフレネル回折波を計算することは現実的ではない．ここでは以下のような奥行き情報を保持した投影像のフレネル回折波[8]を計算する．

Fig. 2に示すような3次元物体に対する基準面 (x,y) を考え，この平面に3次元物体 $g(x,y,z)$ を光軸上の仮想観測点を消点とした透視投影し，それを投影像 $g_{proj}(x,y)$ とする．このときホログラム面と基準面との距離は z_0 とする．また，$g_{proj}(x,y)$ の各点に対応する $g(x,y,z)$ と基準面 (x,y) との距離を $\delta_z(x,y)$ とし，平面投影像のフレネル積分を考える．ただし，通常の2次元物体のフレネル積分で用いる z の代わりに $z_0+(x,y)$ を使用する．すなわち，このときのホログラム面での回折波は

$$u(x_0,y_0)=\iint g_{proj}(x,y)\exp\left[-\frac{ik}{2\{z_0+\delta_z(x,y)\}}\{(x-x_0)^2+(y-y_0)^2\}\right]\mathrm{d}x\mathrm{d}y \quad (2)$$

となる．フレネル積分はもともと物体の広がりに比して z が十分大きい場合，すなわちフレネル領域を想定しており，$g_{proj}(x,y)$ の各点に対応する $g(x,y,z)$ だけを考えれば式（2）は3次元物体 $g(x,y,z)$ の正確なフレネル回折を表現している．また $\delta_z(x,y)$ は $g_{proj}(x,y)$ に付帯した情報として保持していればよく，任意の値を設定できる利点を有している．また，式（1）では隠面問題は考慮されていないが，式（2）では自動的に隠面問題を回避している．

式（2）で計算される回折波 $u(x_0,y_0)$ の振幅と位相を $u_a(x_0,y_0)$，$u_\phi(x_0,y_0)$ とし，ホログラム記録時の参照波 $r(x_0,y_0)$ の振幅と位相を $r_a(x_0,y_0)$，$r_\phi(x_0,y_0)$ とするとホログラム面での干渉縞強度分布 $I(x_0,y_0)$ は

$$\begin{aligned}I(x_0,y_0)&=|u(x_0,y_0)+r(x_0,y_0)|^2\\&=|u(x_0,y_0)|^2+|r(x_0,y_0)|^2+2u_a(x_0,y_0)\,r_a(x_0,y_0)\\&\quad\cos\{u_\phi(x_0,y_0)-r_\phi(x_0,y_0)\}\end{aligned} \quad (3)$$

となる．ここでは第3項のみを計算しホログラムの振幅透過率とする．

3. JavaSpacesによるフレネル型計算機ホログラムの高速計算のための分散並列処理

JavaSpacesはLindaモデル[6]をもとに開発されたもので，エントリーと呼ばれる型付けされたオブジェクトを格納するタップル空間を提供する．その共有空間に対してエントリーを書き込む，読み取る，取り去る等の操作を行える．計算機Aが計算機Bに対して処理を依頼するメッセージを送信する場合を，タップル空間を用いて実現すると次のようになる．まず，Aが送るメッセージを含むエントリーをタップル空間に書き込む．そして，Bは同一の型情報をもつエントリーを用いて，タップル空間より書き込まれたエントリーを取り去る．この2つの処理により，A

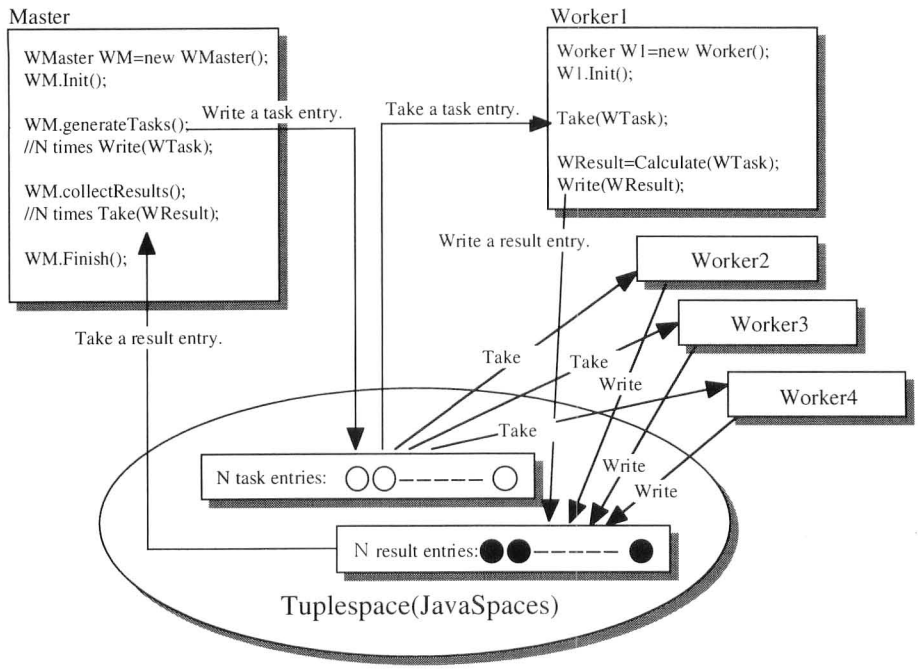

Fig. 3. Replicated-worker pattern with tuplespace.

からBへメッセージが送信されたこととなる．また，タップル空間では，エントリーの型や特定した値が一致すれば，どの計算機でも同じエントリーを取り出すことが可能である．よって，そのメッセージの依頼を処理できる別の計算機B′が，そのエントリーを取り出して処理を行っても構わない．このようにタップル空間は，特定の相手をIPアドレスで特定して通信接続する必要はなく，途中で処理用計算機を増減させるなど柔軟なシステムの記述に適している．

分散並列処理を行うために，ワーカー複製形式[7]と呼ばれる処理形式を用いる．この形式では，処理をマスター部分とワーカー部分とに分ける．マスターは処理全体の制御を行い，仕事を分割してワーカーに配り，ワーカーから処理結果を収集する．ワーカー部分は，割り付けられたデータに対して計算処理を行い，その処理結果をマスターに返すのみである．演算対象とするホログラム面がN行M列のドット面のとき，ワーカーによる計算処理は$N\times M$個の画素数ごとに独立して行える．その処理の分割は，N単位までの分割は行単位で行い，それ以上は，2次元分割すればよい．このワーカーの役割を複数の計算機に行わせることにより，並列処理が実現される．この形式は，大量データに同一処理を加えることが有効な数値計算，画像処理，情報検索などにも応用されている．

Fig. 3は，タップル空間を用いたワーカー複製形式の図示である．マスターは仕事を複数個に分割して，それを仕事内容にもつエントリーをタップル空間に書き込む．そして，タップル空間を監視し，結果をもつエントリーが書き込まれたならば，そのエントリーを取り去る作業を繰り返す．一方，ワーカーはタップル空間を監視し，仕事内容をもつエントリーが書き込まれたならば，そのエントリーをタップル空間より取り去る．そして，その情報をもとに与えられた計算処理を行い，処理結果をもつエントリーとしてタップル空間に書き込む．その結果をもつエントリーを，マスターがすべて収集し終わると，並列処理部分は完了である．タップル空間を用いたワーカー複製形式では，計算途中でも新しくワーカーを追加して，その並列処理能力を向上できる．

4. 分散並列処理の高速化実験結果

4.1 実験内容

実験システムは，本研究室にある計算機環境であり，10 Mbpsの通信速度をもつイーサーネットで接続された複数のパソコンで構成されており，OSはWindows 95，98，NTである．Table 1に実験システムの構成内容を示す．使用したソフトウェアはJavaの開発環境であるJDK 1.2.2とタップル空間等のサービスを提供するJini 1.0である．使用計算機は総数14台であり，マスター用計算機が1台，タップル空間用計算機が1台，ワーカー用計算機が12台である．処理対象とするホログラム面は256行256列のドット面であり，ワーカー用計算機による計算処理は65536個の画素数ごとに独立して行える．その処理の分割は，256単位までの分割は行単位で行い，そ

Table 1. Experimental computers in the parallel system.

Computer	CPU power [MHz]	OS	Role of experimental system
Gateway	266	Windows 98	Master
SOTEC	400	Windows 98	JavaSpaces server
Gateway	266	Windows 98	First worker
Mebius	80	Windows 95	Second worker
Gateway	133	Windows 98	Third worker
Gateway	200	Windows 98	Fourth worker
Gateway	300	Windows NT	Fifth worker
Gateway	300	Windows NT	Sixth worker
SOTEC	333	Windows NT	Seventh worker
SOTEC	400	Windows 98	Eighth worker
SOTEC	400	Windows NT	Ninth worker
SOTEC	400	Windows NT	Tenth worker
Gateway	400	Windows 98	Eleventh worker
Gateway	400	Windows 98	Twelfth worker

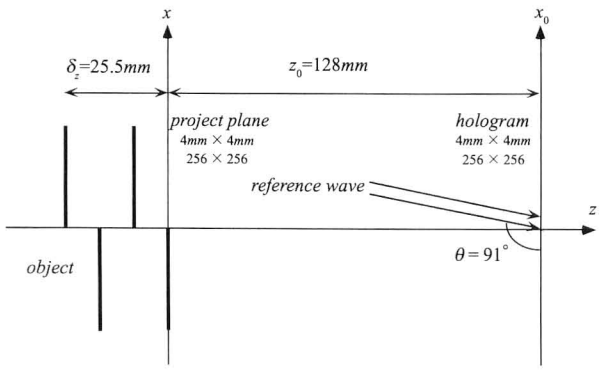

Fig. 4. Overview of the experimental setup.

Fig. 5. Projected image of 3D objects.

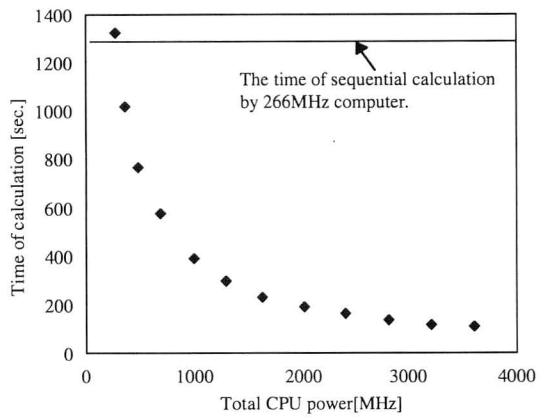

Fig. 6. Calculation time by a parallel system with 256-task partition.

れ以上は，2次元分割した．分割されたホログラム面ごとに3次元物体の回折波を式（2）により計算し，式（3）に従ってそのホログラムの振幅透過率を求めていく．今回の実験で用いたフレネル型ホログラムの記録系をFig. 4に示す．記録する3次元物体は奥行き方向に異なった位置に配置された4つの文字から構成されており，投影面からの位置 $\delta_z(x,y)$ はそれぞれ0.0，8.5，17.0，25.5 mmとした．Fig. 5に使用した3次元物体の投影像を示す．

実験では，タップル空間を用いた分散並列処理の性能について，各計算機のCPU処理能力を合計した値と計算処理の分割数である仕事分割数を中心に調べた．CPU処理能力の合計値が大きくなると，システム全体の計算処理能力は大きくなる．また，仕事分割数の値によって，エントリーがもつ仕事内容の量が変化する．実験システムは，ワーカー用計算機が異なる処理速度をもつことの影響を考慮して，ほぼ中間の処理速度である266 MHzを中心に設定した．この処理速度266 MHzの計算機1台によるホログラム計算の逐次処理時間は1272秒であった．Table 1に示すように，マスター用計算機は266 MHzのものであり，追加するワーカー用計算機の1台目は処理速度266 MHzの計算機として，2台目以降は遅い処理速度の計算機から順に追加した．仕事分割数は4倍刻みで16，64，256，1024，4096，16384個と変化させた．

4.2 実験結果

Fig. 6に，仕事分割数が256個のときにおけるシステム全体の計算処理能力と計算時間の関係を示す．計算処理能力は使用したCPU速度の処理速度を単純に総和したものであり，計算時間は，ワーカー複製形式による分散並列処理の実行時間を含んだ，マスター用計算機が稼動してから最終ホログラムを作成した時点までの時間である．システム全体の計算処理能力が上がれば，計算時間が短くなることがわかる．

Fig. 7に生成されたホログラムのシミュレーターによる再生像を示す．Fig. 7(a)～(d) に $\delta_z(x,y)$ がそれぞれ

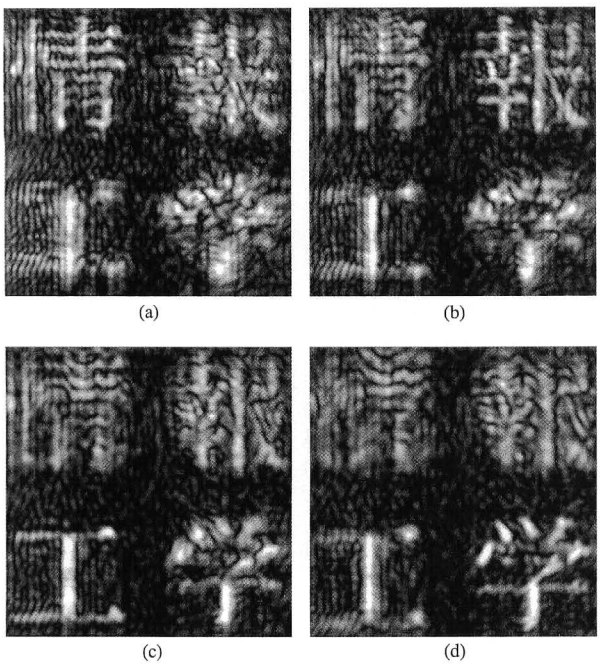

Fig. 7. Images reconstructed by the Fresnel CGH for three-dimensional object. (a) $\delta_z(x,y)=0.0$ mm, (b) $\delta_z(x,y)=8.5$ mm, (c) $\delta_z(x,y)=17.0$ mm, (d) $\delta_z(x,y)=25.5$ mm.

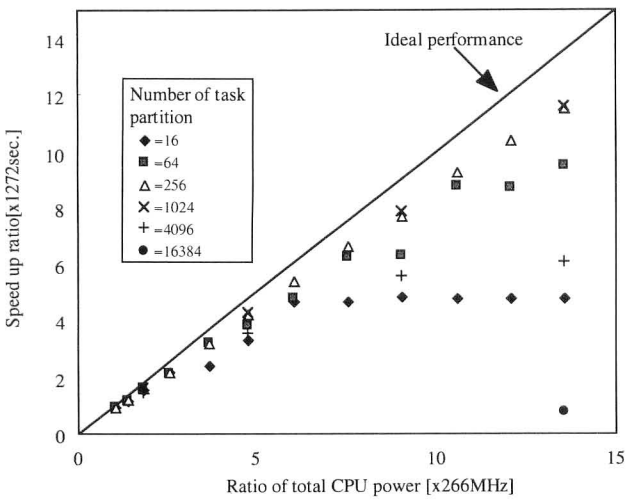

Fig. 8. Relation between the number of task partition and system performance.

0.0, 8.5, 17.0, 25.5 mm の位置での再生像であり，それぞれの文字に焦点が合わせられている状況がわかる．

Fig. 8 に仕事分割数が 16, 64, 256, 1024, 4096, 16384 個ごとの CPU 処理総計比率と速度向上率との関係を示す．CPU 処理総計比率は，用いたワーカー用計算機の CPU 処理速度の合計を，266 MHz で割った値である．速度向上率は，266 MHz の計算機による逐次処理の速度に対する向上率である．仕事分割数が 256 個の場合，CPU 処理総計比率が 13.6 倍のときは，速度向上率は 11.6 倍と

Table 2. Speed of calculation corresponding to the number of task partition.

Number of task partition	Time (s)	Speed up	Loss(%)
16	265.3	4.8	64.7
64	133.8	9.5	30.0
256	110.1	11.6	14.9
1024	109.6	11.6	14.5
4096	208.9	6.1	55.1
16396	1588.2	0.8	94.1

Ideal speed up=13.6

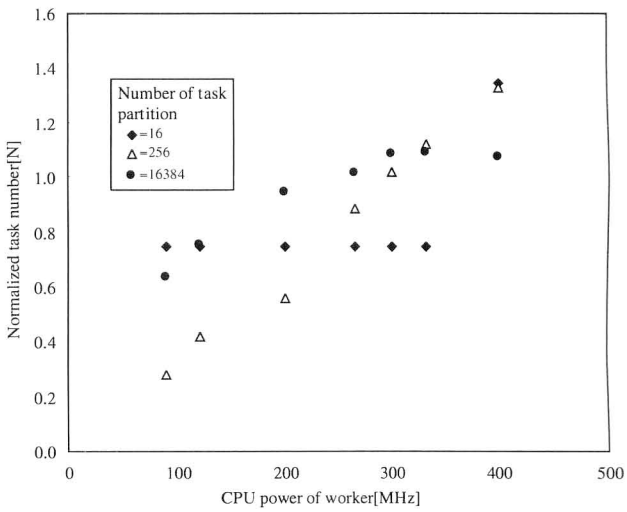

Fig. 9. Load balancing by the number of task partition.

なっているとともに，使用したワーカー用計算機の CPU 処理速度の合計に比例した分散並列処理の効果が出ている．

Fig. 8 より仕事分割数がシステムの処理能力に影響を及ぼすこともわかる．仕事分割数が 256 個と 1024 個のとき，計算処理能力の増大に対応して，システムの処理能力が向上している．一方，仕事分割数がその値に比べて大きい 4096 個のときや小さい 16 個のときは計算処理能力の増大に対して，システムの処理能力が向上していない．

仕事分割によって処理能力の違いが生じる原因を調べるために，最も高い計算処理能力をもつワーカー用計算機 12 台の場合を調べた．Table 2 に，その場合の速度向上率や損失量を示す．損失率は，理想的な速度向上率に実際の速度が及ばない割合を表す．仕事分割数が適当な値でなければ，システムの性能が落ちることがわかる．特に，仕事分割数が 16384 個のときは，94%も性能が落ち，単体で計算したときよりも性能が落ちている．仕事分割数により処理速度が変わる原因を調べるために，計算機の処理能力と負荷分散の関係を調べた．

Fig. 9 にワーカー用計算機の処理能力と仕事処理の関

Table 3. Access time in handling a task.

Number of task partition	Total time			
	Calculation time (ms)	Access time		Access percentage (%)
		Take time (ms)	Write time (ms)	
16	54980.0	16.7	76.7	0.2
64	13808.3	9.4	23.3	0.2
256	3435.6	16.0	21.9	1.1
1024	854.6	27.0	25.1	5.7
4096	213.9	75.2	52.8	37.4
16384	54.6	632.9	83.9	92.9

係を示す．規格化仕事数は，各計算機が実際に処理した仕事数を，ワーカー用計算機1台あたりに期待される仕事数で割った規格値である．ここで，ワーカー用計算機1台あたりに期待される仕事数は，仕事分割数をワーカー用計算機の数で割った値である．CPUの処理能力に応じて規格化仕事数が大きくなれば，適切な負荷分散が行われていると判断できる．よって，仕事分割数が256個のときは負荷分散がうまく行われている．しかしながら，仕事分割数が16個と16384個のときは負荷がうまく分散されていない．

仕事分割数が16個のとき，計算速度が遅くなっているのは，遅い計算機に不相応な負荷がかかっているためである．そのとき，1個の仕事を処理するためにかかる時間を調べると，400 MHzの計算機は平均55秒，80 MHzの計算機は平均290秒であった．よって，マスター用計算機は，遅い計算機の処理が終わるまで，全体処理を終わらせることができない．一方，仕事分割数が多いときに，システム全体の性能が落ちるのは，タップル空間に過度のアクセス負荷がかかるためである．Table 3に，仕事1つを処理するためにかかった時間の内容を示す．これは，400 MHzの計算機1台について調べた結果である．仕事分割数が16384個のとき，ワーカー用計算機が1つの仕事を処理する時間は93%も通信時間に割かれている．その際のシステム内におけるデータ転送速度の遅延を調べると約3 ms/kbyteであった．よって，仕事分割数が多くなると通信の遅延がひどくなるのではなく，タップル空間に対するアクセス負荷が過度になってしまう．その際，タップル空間におけるエントリー操作に時間が費やされ，各ワーカー用計算機の実働時間である計算時間の割合が減少し，システム全体の性能は低下する結果となる．

4.3 システムのフォールトトレラント機能

実験結果より，仕事分割数が少ないとき，処理速度の遅い計算機によってシステム全体の処理能力が遅くなることがわかった．また，ワーカー用計算機が1台でも停止してしまうと，その計算機がもっていた仕事情報が失われて計算が終わらないという問題もあった．

Table 4. Improvement of the system by fault tolerance function.

Number of task partition	Before the improvement	After the improvement
256	186 s	180 s
16	266 s	220 s
Fault tolerance	None	For worker's fault

これらの問題を改善するために，フォールトトレラント機能として，結果の返答が遅い処理を速い計算機に回す機能をワーカー複製形式に追加した．この機能は，故障によりデータがやってこないことは返答が無限に遅いと解釈できるので，ワーカー用計算機の故障排除も行う．マスター用計算機への追加プログラム内容は以下の通りである．タップル空間上に登録した仕事エントリーがすべて消え去ったときは，新たに結果が得られていない仕事エントリーをすべて書き込む．ただし，この処理は一度のみ行う．その後，期待される処理時間内に結果を取得できなければ，再び結果が得られていない仕事エントリーを書き込む．

Table 4に，フォールトトレラント機能によるシステム改善効果を調べた実験結果を示す．実験では，ワーカー用計算機として400 MHzを5台，80 MHzを1台用いて，前述の評価実験と同じ分散並列処理を行った．その結果，仕事分割数が256個のとき計算時間が186秒から180秒へと，仕事分割数が16個のとき，計算時間が266秒から220秒へと短くなり，処理が遅い計算機の影響を減少できた．また，任意のワーカー用計算機1台を処理途中に強制終了させても，無事に計算処理を完了できた．

5. おわりに

計算機ネットワークを用いて，フレネル型計算機ホログラム作成を高速化するための演算方法を検討した．そのために，Java言語で実現されたタップル空間であるJavaSpacesを用い，分割されたデータに対して複数の計算機上で同一処理を加えるワーカー複製形式による分散並列処理を行った．その結果，以下の知見が得られた．

(1) 適切に仕事を分割すれば，さまざまな処理能力の計算機があっても計算機性能にあった応分の負担を行い，システム全体の処理性能を引き出すことができる．

(2) 仕事の分割数が多いときは，タップル空間へ大きな負荷がかかり，システム全体の性能が著しく落ちる．

(3) マスター用計算機に，結果の帰ってこない仕事を投入するフォールトトレラント機能を加えることによ

り，処理の遅いワーカー用計算機の影響を減じることやワーカー用計算機の故障を排除することができる．

今後は，負荷分散を自律的に行う仕組みを検討し，タップル空間への負荷を抑えるとともに，計算機台数に対する拡張性を実現する予定である．

文　献

1) B. R. Brown and A. W. Lohmann: "Complex spatial filtering with binary masks," Appl. Opt., **5** (1966) 967-969.
2) W. H. Lee: "Binary computer-generated holograms," Appl. Opt., **18** (1979) 3661-3669.
3) M. A. Seldowitz, J. P. Allebach and D. W. Sweeney: "Synthesis of digital holograms by direct binary search," Appl. Opt., **26** (1987) 2788-2798.
4) Y. Ichioka, M. Izumi and T. Suzuki: "Scanning halftone plotter and computer-generated continuous-tone hologram," Appl. Opt., **10** (1971) 403-411.
5) 高野邦彦，小山　健，佐藤甲癸："ネットワークによるホログラム計算の高速化に関する検討"，第5回 HODIC 講演会講演論文集 (1997) pp. 5-8.
6) D. Gelernter: "Generative communication in Linda," ACM Trans. Program. Lang. Syst., **7** (1985) 80-112.
7) E. Freeman, S. Hupfer and K. Arnold: *JavaSpaces (TM) Principles, Patterns, and Practices* (Addison Wesley, Massachusetts, 1999).
8) K. Mori, R. Takane and R. Sato: "Computer-generated Fresnel hologram for three dimensional object," Proc. SPIE, **2778** (1996) 573-574.