A-016

MPIを用いたPCクラスタ上での自己安定分散相互排除アルゴリズムの実験的評価

Evaluation of Self-Stabilizing Distributed Mutual Exclusion Algorithms on PC Cluster using MPI

近藤 紹弘 † Tsuguhiro Kondo
角川 裕次 † Hirotsugu Kakugawa

1. はじめに

近年, PCやネットワーク環境の高性能化が進み, これまではスーパーコンピュータで行っていた大規模計算をPCクラスタを用いて実行する事が可能になっている. さらに, PCクラスタは大規模計算だけでなく, 信頼性の向上やコストの削減と言った多くの利点があり, 企業, 大学等でますます需要が高まってきている. PCクラスタを稼働させるには, 資源割り当て等の相互排除を要する場面が多くある. 相互排除とは, 多くの競合するプロセスの中から1つのプロセスを選出し, そのプロセスに対して特別な権利を与える競合解消の事である. 大きな計算になるとPCの台数, 計算時間ともに増大するので故障が起こる可能性が大きくなる. したがって相互排除アルゴリズムは, PCの一時故障が発生しても自己回復することのできる自己安定アルゴリズムであることが望ましい. 自己安定アルゴリズムは, ネットワークの初期状態に何も仮定せずに有限時間内に問題を解くことができる分散アルゴリズムである. この特徴より一時故障が生じたときの状態を初期状態と考えると, 有限時間内に安定状態に自己回復する事ができる. 自己安定アルゴリズムの動作は, 安定するまでと安定した後の二つに分けて考えることができる. 安定後の動作については解析が比較的容易だが, 安定するまでの動作は解析が難しい. そこで, 本研究ではクラスタ上で実行される自己安定相互排除アルゴリズムの振る舞いを特に初期状態から安定状態に達するまでを重点的に評価する.

2. 実験

2.1 実験環境

PCクラスタではプロセス一つがPC一台にあたりPC間の通信リンクはイーサネット等を使用する. そして, PC間の情報交換はメッセージ通信で行われるが, そのインターフェースとしてMPI(Message Passing Interface) 等が使用される. 本研究の実験環境は, OSにLinux kernel 2.4.18, MPI実装パッケージにMPICH Ver 1.2.4, 通信リンクに100BASE-TXイーサネットをそれぞれ使用し, クラスタシステムタイプは, CPUの種類, 搭載メモリ容量等の性能が異なった様々なPCから構成されるヘテロクラスタとする. 各PCの処理速度の違いはMPIによるメッセージ通信のオーバヘッドと比べると無視できる程小さいので, 性能の差が結果に影響を及ぼすことはない.

2.2 評価アルゴリズム

評価の対象とするアルゴリズムは, Nesterenkoらにより提案された [3](以下 DSSM) と角川らにより提案された [1](以下 TSSM) を用いる. 双方ともコーラムを用いて相互排除を行う自己安定アルゴリズムである. コーラムとは, U をプロセスの集合とした時, 以下の2条件を満たす空でない U の部分集合 Q_i であり, コーラム Q_i の集合 $\{Q_1, Q_2, \ldots, Q_N\}$ をコータリと呼ぶ.

1. 全ての対 i と $j(1 \leq i, j \leq N)$ に対して, $Q_i \cap Q_j \neq \emptyset$.
2. 全ての対 i と $j(1 \leq i, j \leq N, i \neq j)$ に対して, $Q_i \not\subseteq Q_j$.

これらの相互排除アルゴリズムの動作概要は, 臨界領域 (以下 CS) に進入を試みるプロセスは自分が属しているコーラムの全てのプロセスに許可を求め, 全てのプロセスから許可を得た時のみにCSを実行できるというものである. コーラムの構造上この動作で相互排除が可能となる. 文献 [1] のアルゴリズムでの通信モデルと実行モデルは実際のPCクラスタとは異なるため, 文献 [2] で提案された手法でモデル変換を行い実装した.

2.3 評価内容

アルゴリズムDSSMとTSSMにおいて以下の内容をシミュレーションにより評価する.

1. 安定までに必要なメッセージ数

いづれのアルゴリズムも安定時間の上限は示されてはいるものの, 故障の度合いと安定時間の関係については知られていない. したがって, 今回のシミュレーションでは, アルゴリズムの実行中に故障を起こして, その故障状態から安定状態に達するまでに必要なメッセージ数を以下のパラメータで評価する. シミュレーション中に起こす故障は, (1) 通信リンク故障：MPIによるメッセージ送信の損失, 誤り (2)PC故障：プログラムの実行変数のリセット, の2種類とする.

- PCの故障台数： 故障が起こるPCの台数を1台, 全体の25%, 50%, 75%, ALLと変えて評価
- 故障のタイプ：故障のタイプを通信リンク故障, PC故障と変えて評価

2. 相互排除効率

安定後の相互排除効率をCS要求発生確率毎に, 一回のCS実行に必要なメッセージ数で評価する.

2.4 プロセスの状態検出

各プロセスの故障状態, 安定状態の検出は次のようにする.

1. 安定時における, プロセスの取りうる状態を実行変数の値により決めておき, その状態間の遷移規則も決めておく.

†広島大学大学院工学研究科

2. プログラム実行中に各プロセスの状態遷移を監視し, 安定時の遷移規則どおりに状態遷移を繰り返しているときは, 安定状態と判断し, 遷移規則から外れているときは故障状態と判断する.

3. 結果

表 1: 故障から安定までのメッセージ数：DSSM

評価項目		PC 台数				
		4	9	16	25	36
故障台数	1	22	73	258	485	952
	25%	60	174	322	778	1592
	50%	50	127	348	929	1503
	75%	44	190	422	962	1545
	ALL	61	193	409	834	1769
故障タイプ	リンク	19	35	67	344	826
	PC	70	207	427	941	1483

表 2: 故障から安定までのメッセージ数：TSSM

評価項目		PC 台数				
		4	9	16	25	36
故障台数	1	82	222	427	662	1154
	25%	102	312	841	2036	2293
	50%	134	499	949	1979	3482
	75%	234	928	1374	3144	3959
	ALL	215	978	1784	4204	5097
故障タイプ	リンク	23	161	555	643	1492
	PC	65	585	1694	3404	6223

表 3: 安定後の相互排除効率 (Msg/CS)：DSSM

	PC 台数	4	9	16	24	36
CS 要求確率	0.001	24	24	24	24	24
	0.01	25	27	29	32	34
	0.1	32	32	34	35	37
	1	33	34	34	36	38

表 4: 安定後の相互排除効率 (Msg/CS)：TSSM

	PC 台数	4	9	16	24	36
CS 要求確率	0.001	430	456	488	582	821
	0.01	432	454	481	582	809
	0.1	418	462	485	583	822
	1	414	461	489	575	823

3.1 故障 PC の台数に関して

DSSM においては表 1 より, 故障 PC が 1 台の時でも安定までにある程度のメッセージ数は必要になっているが, 全故障時に比べると半分程度のメッセージ数で安定することが出来ている. このようになる理由は, 故障 PC が安定状態に回復するためには, その故障 PC が安定時に通信する全ての PC に自分の現状態を送信する必要があり, 1 台故障時は全故障時に比べると, その現状態を送信しなければいけない PC 数が少ない為である.

TSSM においては表 2 より, 安定するまでのメッセージ数は故障 PC の台数にほぼ比例しており, 故障 PC が 1 台の時は全故障時に比べるとかなり少ないメッセージ数で安定している.

実際の PC クラスタ稼働中の PC 故障は, 同時に発生するのは 1,2 台程度なので, PC クラスタの運用においては, こういった 1,2 台故障 (小故障) から安定までの動作が重要である. 今回のシミュレーションより, DSSM, TSSM 共に小故障時は安定が早いという結果が出たのでこれらのアルゴリズムは PC クラスタ上で有効に働く事が確認出来た.

3.2 故障タイプに関して

表 1, 表 2 より, 故障タイプに関しては DSSM,TSSM 共に, PC 故障に比べるとリンク故障の方がだいぶん早く安定することができている. これは, PC 故障が起こると, プログラムの実行変数, プログラムカウンター等がリセットされて PC の状態が大きく混乱するのに対して, リンク故障の場合は, メッセージ誤りではそのメッセージの破棄, メッセージ損失ではメッセージの再送を行うのでシステムが大きく混乱することはない為である.

PC クラスタの運用時において, リンクの一時的な故障は PC 故障より発生しやすいので, これらのアルゴリズムのリンク故障は PC 故障より安定がはやいといった特徴は, PC クラスタ運用にあたり有効であると思われる.

3.3 相互排除効率に関して

DSSM においては表 3 より, (1)CS 要求確率が高くなる程メッセージ数は増加する. (2) クラスタを組む PC 台数が多くなってもメッセージ数はあまり変わらない. などの事が言える. このため DSSM は, PC 台数の多い大規模なクラスタにおいて効率良く相互排除を行える事が分かった.

TSSM においては表 4 より, (1)CS 要求確率が高くなっても一定のメッセージ数で相互排除が行える. (2) クラスタを組む PC 台数が多くなるとメッセージ数は増加する. などの事が言える.

TSSM は 2.2 節で説明したとおり, 自己安定アルゴリズムを設計と検証の容易な通信モデルで設計し, モデル変換を用いることにより PC クラスタ上での自己安定アルゴリズムとしている. つまり, 設計と検証の容易さと引換えに実行時のオーバヘッドが大きくなっている.

4. おわりに

今回の研究では, 二つの自己安定相互排除アルゴリズムを MPI を用いて実際の分散システム上でシミュレーションし, 様々なパラメータのもとでアルゴリズムの振舞いを確認した.

今後の課題としては, コーラム及びネットワーク形状の違いによるアルゴリズムの評価が挙げられる.

参考文献

[1] H. Kakugawa, M. Mizuno, and M. Nesterenko. Development of self-stabilizing distributed algorithms using transformation: Case studies. *In Proc. of the Third Workshop on Self-Stabilization (WSS'97)*, pp. 16–30, 1997.

[2] M. Mizuno and H. Kakugawa. A timestamp based transformation of self-stabilizing programs for distributed computing environments. *In Workshop on Distributed Algorithms (WDAG'96)*, pp. 304–321, 1996.

[3] M. Nesterenko and M. Mizuno. A quorum-based self-stabilizing distributed mutual exclusion algorithm. *Journal of Parallel and Distributed Computing*, Vol. 62, pp. 284–305, 2002.