クライアントサーバシステム構築支援

2R－9

野澤 幸輝　藤巻 昇　岩田 誠司　小尾 俊之
株式会社 東芝 研究開発センター システム・ソフトウェア生産技術研究所

1　はじめに

本稿では、クライアントサーバシステム (以下 CSS) を構築する際に、プロセスやデータを、ネットワーク上にどのように分散配置させれば良いのかについて、その決定に対する支援を行う、CASE ツールについて述べる。

2　目的

CSS を構築するには、システムに課せられた技術的／経済的／ポリシー等の制約を考慮しつつ、良好な性能を発揮するように、プロセスやデータを配置しなければならない。しかし、システムの大規模化やアーキテクチャの複雑化により、有効な配置を得るのは難しい。

この問題を解決するひとつの方法として、CSS 運用時のパフォーマンス情報から、配置の有効性を開発者が判断し、GUI を用いた手段によって配置を容易に変えられるという方法を、多くのツールが採用してきている [1]。

しかしこれらのツールは、配置を整理するという手間に対する支援はしているが、配置案を提示するという思考に対する支援まではしていない。そこで本稿では、設計工程における配置問題において、技術的／経済的／ポリシー等の制約を考慮しつつ、もっとも有効な配置を得るための思考作業を支援するツール像を提案することを、目的とする。

3　要件

本稿ではツールの利用という観点から、ツールの使い勝手を第一義とする。通常このようなツールの場合には、得られた配置がどれだけ有効なものであるかという最適性が重要視されがちであるが、以下に示すように、最適性を追求すると使い勝手が損なわれてしまう。

- 最適性を追及すると、配置にかかる時間が長くなる。
- 最適性を追及すると、配置に必要な情報が多くなる。

そこで本稿では、次の２点を満たすようにした上で、より有効な解を得られるツールを考える。

インタラクティブ性　CSS の構築においては、プロトタイピング手法を取ることが一般的になってきている。そのためには、入力に対して結果が返ってくるまでの時間が、短くなければならない。

運用性　ツールの恩恵を受けるために、多くの入力が必要になってしまうようなことが、あってはならない。

4　実現

まずはじめに、配置問題について定義する。図 1は、本稿から見た CSS の世界を表現する ER 図であり、本稿のツールへの入力となる。このように簡単に表現するのは、

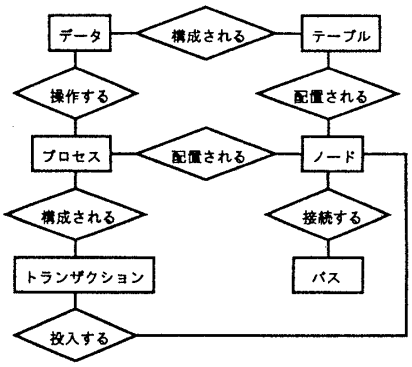

図 1: CSS を表現する ER 図

3. で示した要件のひとつ、運用性を確保するためである。

図 1の定義のもと、プロセスおよびテーブルを重複を許してノードに配置していく。この配置結果が本稿のツールの出力となる。ここで本稿では、表 1に示すようなトレードオフの関係にある評価項目を設定し、配置の有効性を評価することとした。

本稿では、組合せ最適化問題である上記配置問題を、準最適化アルゴリズムのひとつである、遺伝的アルゴリズム (以下 GA) を用いて解くものとした。準最適化アルゴリズムは、最適性を犠牲にする代わりに、解を得るまでの時間を短くできる。よって GA を採用することにより、3. で示したもうひとつの要件、インタラクティブ性を確保することができる。

Client/Server System Construction Support
Yukiteru Nozawa, Noboru Fujimaki, Seiji Iwata and Toshiyuki Obi
Systems and Software Engineering Laboratory
Research and Development Center, TOSHIBA Corp.
3-22 Kata, Fuchu, Tokyo

配置対象	評価項目						
	性能		信頼性	セキュリティ		整合性	コスト
	*1	*2		*3	*4		
プロセス	+			-			
テーブル*5	+				-		
テーブル*6	+	-	+		-		-
テーブル*7	+	+	+		-	-	-

*1 通信ボトルネック　*4 テーブル　*7 実体の非同期コピー
*2 同期更新ボトルネック　*5 実体
*3 プロセス　*6 実体の同期コピー
表中 + は、配置対象が多く配置されるほど、評価項目を満足し、
- は、配置対象が多く配置されるほど、評価項目を損なうことを意味する。

表 1: 評価項目間のトレードオフ

5 評価

4. で示した GA を用いた支援ツールのプロトタイプを用意し、例題に適用した結果を示す。

例題として、図 2に示すようなノードおよびパスに対して、図 3に示すようなトランザクション／プロセス／テーブル[1]を、性能とコストを優先して配置させる問題を取り上げる。

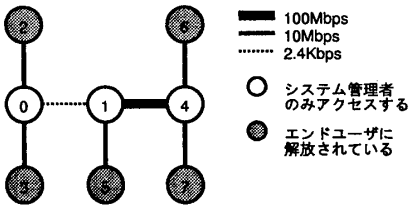

図 2: 例題におけるノードおよびパス

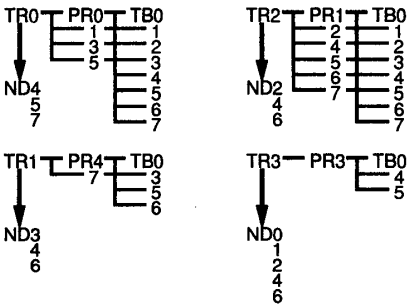

図 3: 例題におけるトランザクション／プロセス／テーブルおよびノード

例題に対する配置結果を表す図 4から、以下のことがわかる。

[1]図 3においては、トランザクション（図中 TR）を構成するプロセス（図中 PR）とそれがアクセスするテーブル（図中 TB）の関係、およびそのトランザクションが投入されるノード（図中 ND）の関係が示されている。

ノード	プロセス 0	1	2	3	4	5	6	7	テーブル 0	1	2	3	4	5	6	7
0	Y	Y	Y	Y	Y	Y	Y	Y	A	A	E	A	A	S	A	E
1	N	Y	N	Y	Y	Y	N	Y	A	E	N	N	A	S	A	A
2	Y	Y	Y	Y	Y	Y	Y	Y	S	A	S	A	A	A	E	A
3	Y	N	N	Y	Y	N	N	N	S	N	N	S	A	E	S	N
4	Y	Y	Y	Y	N	Y	Y	Y	E	A	A	A	E	A	A	A
5	Y	Y	Y	N	N	N	Y	Y	A	N	A	A	N	A	A	N
6	N	N	N	N	N	Y	Y	N	A	N	N	E	A	N	A	N
7	Y	N	Y	N	N	Y	Y	N	N	N	N	N	N	A	N	N

Y 配置する　E 実体を配置する　A 非同期コピーを配置する
N 配置しない　S 同期コピーを配置する　N 配置しない

図 4: 例題に対して性能とコストを強調した配置結果

- ノード 4,5,7 から投入されるトランザクション 0 でのみ使われるプロセス 0 は、性能を向上させるためにノード 4,5,7 に配置されており、コストを抑えるためにノード 1,6 には配置されていない。
- すべてのトランザクションでアクセスされるテーブル 0 は、性能を向上させるためにノード 7 以外のすべてに配置され、トランザクション 0,2 でしかアクセスされないテーブル 1 はコストを抑えるために半分のノードにしか配置されていない。
- 性能を向上させるために、全体的に非同期コピーによるテーブルの複製が目立つ。

本稿では図 4に示した結果を、表 2のリソースのもとで得ることができた。

マシン	SparcClassic	個体数	64
メモリ	32M	探索打ち切り	約 1 分 (1000 世代)

表 2: 例題を解くのに要したリソース

以上からわかるように、 4. で示した実現方法によるツールが、 3. で示した要件を満たしながら[2]、かつ有効な配置結果を得られるものであることが分かった。

6 おわりに

今後、実適用可能なツールとして仕様を詰め、ツールとしての完成度の向上を図っていく。

参考文献

[1] “大規模トランザクション対応の C/S アプリ開発・運用ツール”, ネットワークコンピューティング, No.73, pp.99-103, Apr. 1995

[2]インタラクティブ性は GA により、運用性は図 1に示した少ない入力により、獲得されている 。