マルチメディア通信と分散処理　52－25
（1991．9．26）

並行オブジェクト指向言語 superC^2 の設計と実現

今井　功　　佐藤　文明　　勝山　光太郎　　水野　忠則

三菱電機（株）情報電子研究所

あらまし　我々は，並行オブジェクト指向言語 superC^2 の開発を行なった．これは，複数の計算機がネットワークによって接続されている分散環境上で，作動するアプリケーション・ソフトウェアの開発を行なうために言語レベルで提供したものである．superC^2 では従来，通信ソフトウェア用に開発したオブジェクト指向言語 superC の性質を完全に受け継ぎ，更に並行処理動作を可能としている．本論文では，superC^2 の設計構想とシステムの実現方式について報告するとともに，分散システムに発生する問題を解決するアルゴリズムの設計をもとに実用的な面からの評価について論じている．

Design and Inplementation of The Concurrent Object-Oriented Language superC^2

Isao IMAI　Fumiaki SATO　Kotaro KATSUYAMA　Tadanori MIZUNO

Computer & Information Systems Laboratory,
Mitsubishi Electric Corporation.

5-1-1 Ofuna, Kamakura, Kanagawa 247, Japan

Abstract　This paper describes concurrent object-oriented language superC ^{2}. This provides programming level interface for development of application software that work on distributed environment that some computers are connected by network. We have developed object-oriented language superC for communication software development. SuperC ^{2} inherits superC's nature, and is extended the syntax to describe concurrent processing. This paper reports for superC ^{2}'s design conception and implementation. Furthermore, this paper describes the estimation from the practical viewpoint with an example of design for distributed dead-lock detection algorithm.

1 はじめに

ワークステーション (WS) およびネットワークの高速化，高性能化によって，複数の WS をネットワークで接続し，一つの仕事を分散して実行する分散処理システムが可能となってきた．

分散処理システムは並行処理を可能とし，全体的な処理のスピードアップ化や，分散システムの構成要素間で資源の共有ができるといった効果が期待できる．我々は分散処理システム上でのアプリケーション開発に対応するものとして，オブジェクト指向言語 superC[1] に並行動作機能を拡張した並行オブジェクト指向言語 superC^2[2][3] の開発を行なった．superC^2 は，並行処理プログラミングのし易さと，分散処理システムとの親和性の評価を目的としている．

本論文では，はじめに並行オブジェクト指向言語 superC^2 の設計として superC から拡張する基本方式について述べる．続いて，システムの実現方式として superC^2 の言語仕様と動作環境等について述べる．最後に superC^2 を実際のアルゴリズムに適用しシステム上に実装することによって，有効性の評価を行う．

2 superC^2 の設計

2.1 superC の特徴

superC は，従来のシステム記述用言語である C 言語に，オブジェクト指向の概念を付加した言語であり，通信ソフトウェアの開発効率化を目的としている．superC の主な特徴を以下に挙げる．

(1) **継承機能**：第一の特徴として superC は，インスタンスを生成する際に上位クラス（スーパークラス）の性質を受け継ぐことができる．このことから，開発ステップ数の縮小化が期待できる．ただし，継承はメソッドに限られ，インスタンス変数は継承できない．

(2) **情報隠蔽**：第二の特徴として superC は，強い情報の隠蔽能力を持つ．これによって superC は，ソフトウェア構成要素間の依存関係を少なくし，ソフトウェア・システムの信頼性を確保する．また，他の構成要素に影響することなく修正を行なうことができる．

(3) **メソッドの動的束縛，静的束縛**：メッセージ・パッシングには，プログラムの実行時にメソッドを検索し実行する動的束縛と，コンパイル時に実行すべきメソッドを決定する静的束縛の 2 つの方法がある．通常のオブジェクト指向言語で用いている動的束縛は，メッセージに反応するメソッドが実行時に検索し実行されるので，オブジェクトの独立性を考えた場合重要であるとはいえるが，その反面実行速度の低下を生むといった短所が見受けられる．これに対して静的束縛は，設計者はメッセージを送るとき，どのオブジェクトに対して送るかということを考えなければならないが，実行時において速度低下を防ぐことができる．前者はプロトタイプ用で，後者は実際のシステム開発用であるといえる．superC は，この 2 つの方法を，その用途によって自由に使い分けることができる．

(4) **分割コンパイル**：superC は，プログラム開発の効率化を計るために，大規模なシステム設計において有効となる分割コンパイルをサポートしている．

(5) **C 言語との互換性・混合記述**：superC の言語仕様は C 言語を包含する形をとっており，完全な C 言語のスーパーセットになっている．また，関数呼び出しとメッセージ・パッシングの混在，関数定義とクラス定義の混在を可能とし，実行速度が要求される部分は，C の関数呼び出しによって記述することが可能である．

(6) **移植性**：superC のプリコンパイラの生成コードは C 言語で記述されており，またプリコンパイラ自体も C 言語で記述されている．よって，C コンパイラがあれば，どのマシンにおいても移植することが可能である．

2.2 superC の言語仕様

superC のプログラム例を図 1 に示し，各構文について説明する．

```
1   class samp root {
2       int x;
3   }
4   {
5       add(a,b)
6       int a, b;
7       {
8           x = a+b;
9       }
10      pri(){
11          printf("add -> %d ¥n",x);
12      }
13  }
14
15
16  main(){
17      object  samp0bj;
18
19      samp0bj = (object)samp<new>;
20      samp0bj<add:1,2>;
21
22      samp0bj<pri>;
23
24      samp0bj<purge>;
25  }
```

図 1　superC のプログラム例

図 1 において，line1 から line13 までがクラスの宣言である．line1 の samp は，クラス名を示している．クラス名の後に続いて、スーパークラス（そのクラスより上位のクラスを言う）を指定する。本例は、スーパークラスが root クラスであることを示している。root クラスとは、クラス構造の最上位にあるクラスである。

line2 は，インスタンス変数の宣言である．ここで宣言された変数は，クラス内の全てのメソッドから変数の内容を参照することができる．

line5 から line12 までがメソッドの定義部である．定義方法は，C言語の関数定義と同じであるが，メソッド内には，メッセージ・パッシングを記述することができる．

line19 では，予約語 "new" を用いてクラス samp からインスタンス sampObj を生成している．

line20 および line22 は，sampObj に対する動的束縛によるメッセージ・パッシングの例である．< >内には，実行すべきメソッド名と ":" の後に，メソッドに渡す引き数を記述する．式前部に "クラス名::" を付加することにより，静的束縛によるメッセージ・パッシングが可能となる．

line24 では，予約語 "purge" によってインスタンス sampObj が消去される．

2.3 並行処理機能拡張の基本方式

superC^2 の設計目的は，通信・同期といった部分を設計者から隠蔽することにより，分散処理システムの開発の容易化・統一化を計ることである．また，並行処理で問題となる並行動作部分の抽出といった，複雑な問題を強く意識せずに設計することが可能となる．以下に superC^2 の基本的な特徴について述べる．

(1) **処理の単位**：我々が対象にしているのは，LAN (Local Area Network) 上で高速なワークステーションのアプリケーションソフトウェアが，相互に通信しながら協調して作業を行なうシステムである．そのため，並行動作の単位はアレイプロセッサ上での処理対象のような小単位ではなく，粒度の粗いものを意識している．superC^2 では，この並行動作の単位をオブジェクト指向との整合性をとるために superC の処理の単位でもあるオブジェクトとし，異なったオブジェクトのメソッドは並行に実行可能とする．我々はこれを，並行オブジェクトと呼ぶ．これに対し，従来の superC の処理単位を "逐次オブジェクト" と呼ぶ．

superC^2 は，逐次と並行の2種類のオブジェクトを持つ．

並列のオブジェクトを作成せずに全て逐次型のオブジェクトでプログラムを作成した場合は，通常の superC で作成したものと同様の動作になる．

(2) **並列性と同期**：並行処理にあるオブジェクト間のメッセージ・パッシングには，並列オブジェクト指向言語 ABCL/1 で検討された，3種類の方式がある [4]．

superC^2 ではこのうち，過去型と呼ばれる，非同期式のメッセージ通信方式 (図2) を採用する．以下は，本方式を採用した理由である．

(a) 同期通信より並行性が高く，シンプルなプログラムが記述できる．

(b) 同期通信が必要なときは，非同期通信のみによって模倣することができる．

(c) ＲＰＣのような受信側の応答を待つ必要がなく，デッドロックを起こすことがない．

また，オブジェクト指向型の統一性を保つため，同期はメッセージによって行ない，同期機構は用意しない．

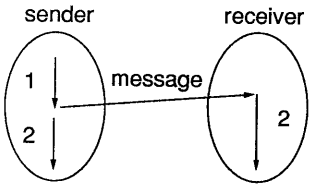

図2 メッセージ・パッシング方式

(3) **互換性**：superC^2 は，superC の言語仕様に対して完全に上位互換を保っている．これによって以下の利点が得られる．

(a) オブジェクトに対する統一的なアクセス方法を提供することによって，superC^2 の処理系を使用する場合でも，従来の superC によってプログラミングされた部分を変更する必要がない．

(b) プロセス間通信をメッセージ通信方式に隠蔽することによって，ネットワークやプロセスを強く意識することなく設計できる．

3 superC^2 の実現方式

3.1 言語仕様

superC^2 のプログラム例を図3に示し，各構文について説明する．

図3において，line1 から line29 までがクラスの宣言である．宣言方法は，前述 (2.2) の方法と変わらない．line9,24 は，自分自身のクラス内にメッセージ・パッシングする方法である．

line1 のクラス aClass および line16 のクラス bClass が別ファイルであるときは，line32 のように extrenclass 宣言する必要がある．

line37,38 は，並行オブジェクトの生成構文である．生成方法は，逐次型の構文 (図1 line19 参照) と変化がないが，並行オブジェクトは，予約語 "parl" を修飾することによって生成される．これによって，指定したリモートホストにプロセスとして生成される．

```
1  class aClass root {
2      int x;
3  }
4  {
5      add(a,b)
6      int a, b;
7      {
8          x = a+b;
9          self<pri>;
10     }
11     pri(){
12         printf("add->%d ¥n",x);
13     }
14 }
15
16 class bClass root {
17     int x;
18 }
19 {
20     sub(a,b)
21     int a, b;
22     {
23         x = a-b;
24         self<pri>;
25     }
26     pri(){
27         printf("sub->%d ¥n",x);
28     }
29 }
30
31
32 externclass aClass, bClass;
33
34 main(){
35     object  aObj, bObj;
36
37     aObj=parl::(object)aClass<new>;
38     bObj=parl::(object)bClass<new>;
39
40     parl::aObj<add:5,3>;
41     parl::bObj<sub:5,3>;
42
43     aObj<purge>;
44     bObj<purge>;
45 }
```

図3　superC^2 のプログラム例

line40,41 は，並行処理メッセージ・パッシングである．superC^2 では，メッセージ式が逐次処理と並行処理の 2 方式が用意されている．逐次処理メッセージとは，従来の superC のメッセージ・パッシングである（図 1 line20,22 参照）．このタイプのメッセージは，戻り値を持つことができる．

line40,41 の並行処理メッセージ・パッシングは，その式が実行されると，送り先の処理の終了を待つ必要がなく，制御が戻ってくるタイプである．このタイプのメッセージは，返り値を持つことができない．これは生成構文と同様，予約語 "parl" を修飾する．並行処理メッセージには，実装上の制約があり，例えば引数の値がポインタの場合は，サイズを指定をしなければならない．

line43,44 では superC 同様，予約語 "purge" によってリモートホスト上のプロセスが消去される．

3.2　プリコンパイラ

superC^2 のプリコンパイラは，superC と同様，C の中間コードを生成する．このコード中には，並行動作をするオブジェクトが一つのロードモジュールとして動作できるように，main 関数と，ソケットからの入力待ちのプログラムが付加される．並行動作のメッセージ・パッシングはプロセス間通信に変換する．

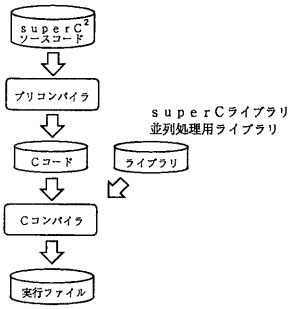

図4　コンパイラの構成

3.3　並行オブジェクト指向モデル

分散処理における並行動作の基本的な計算モデルとしては，Concurrent Process モデル，CSP モデル，Actor モデル等 [2] がある．ここでは，実行主体が動的に生成消滅可能で，メッセージが基本的に非同期である点が 2.2 で述べた対象に適していることから Actor モデルを採用した．Actor モデルを利用した並行オブジェクト指向には，ABCL/1，CA-BLE[6] 等がある．上記の動作モデルを実現する環境としては，以下の機能が実現されている必要がある．

(1) 並行オブジェクトの生成機能

(2) 並行オブジェクトの管理機能

(3) 並行オブジェクトの消去機能

3.4　動作環境

superC^2 を並行動作させるための環境として，オブジェクトマネージャを開発した．オブジェクトマネージャ（OM）の基本動作を図 5 に示す．OM は，図 5 に示すように，並行オブジェクトから他の並行オブジェクトのインスタンス生成の要求を受け付ける (1)．要求されたオブジェクトが，異なる計算機にある場合，OM はその計算機にインスタンス生成要求を転送する (2)．生成要求を受けた OM は，自分の計算機上にインスタンスをプロセスとして生成すると (3)．そのインスタンスにメッセージを送るための情報をインスタンス識別情報として要求側のマネージャに返送する (4)．要求側の OM は，要求元の並行オブジェクトにインスタンス情報を返す (5)．リモート間のメッセージ通信は，全て OM 経由で行う (6)(7)(8)．OM は，メッセージの宛先よりそれに対応するプロセスに対してメッセージを配送する．OM とインスタンスオブジェクトとは，ＴＣＰ／ＩＰプロトコルでコネクション通信をしている．

OM は，クラスの存在するマシンを管理するクラステーブルと，オブジェクトの生成情報を管理するオブジェクト

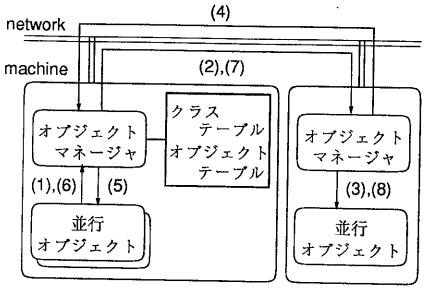

図5 オブジェクトマネージャの機能

テーブルを持っている．オブジェクト生成要求のクラスが，クラステーブルでローカル以外のマシンに対応していれば，OMはテーブルに記述されたリモートマシン上のOMと通信し，オブジェクトの生成を要求する．

また，オブジェクトテーブルでは，生成したオブジェクトのプロセスID，生成元のオブジェクトのプロセスID，生成オブジェクトのポートIDが管理されている．オブジェクトの消去要求がOMに到着すると，OMは対応するプロセスにｋｉｌｌシグナルを送る．また，OMは一定時間間隔で生成元のオブジェクトが生存しているかどうかの確認を行っており，プロセスが存在しない場合はそのオブジェクトをｋｉｌｌする．これは，オブジェクトのガベージコレクション（ＧＣ）に対応しているが，現在は複雑な参照関係までは考慮していない．また，OMが管理するマシン（クラステーブルで管理されている）には，必ずOMが動作していることが前提となっている．

4 分散アルゴリズムへの適用例

本章では，superC^{2}を分散型のアルゴリズムの実現に適用することにより，superC^{2}の有効性の評価を行ない，実用上の問題点について考察する．

4.1 評価システムの概要

オブジェクト指向分散システムで生じる問題の一つに，並列関係にあるオブジェクト間のリソース獲得問題がある．

オブジェクトがリソースを獲得する際，オブジェクト間のスケジューリングを行なうために，獲得するリソースに対してロックを行なう必要がある．しかし，逆にこれはシステムに対してデッドロックをおこす要因ともなる．

これを解消する方法として，Marina Roesler and Walter A. Burkhard が提唱した，オブジェクト指向分散システムにおけるデッドロックの解消アルゴリズム[7]がある．本論文では，このアルゴリズムを分散システム上でモデル化した（図6参照）．

このモデル上で生じるデッドロック状態を解決する手段が，今回用いた解消アルゴリズムである．superC^{2}の評価として本アルゴリズムを適用した理由を以下に示す．

(1) ネットワークで接続された各ノードが，ローカルな情報とメッセージのやり取りのみで，全体として１つの問題を協調し合いながら解決するアルゴリズムで，メッセージ駆動による実現が基本となっている．

(2) メッセージ待ちによるデッドロックが生じないという，superC^{2}のメッセージ通信のメカニズムが，本アルゴリズムと親和性がとれる．

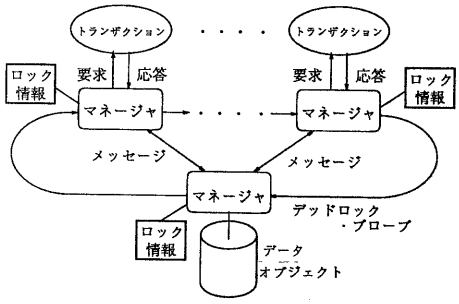

図6 オブジェクト指向分散システム

4.2 superC^{2}による設計

本システムは，トランザクション(T)，トランザクション・マネージャ(TM)，データオブジェクト・マネージャ(OM)，データオブジェクトによって構成されている．各々の役割についてみてみると，Tは，TMを介してOMに対しロックもしくはアンロックの要求を送る．

表1　OMの構造

要素	内容
Holders Pool	オブジェクトにロックをかけているトランザクションの情報
Conflict Pool	オブジェクトに出している要求が満足されていないトランザクションの情報
Probe Queue	デッドロック検出，解消の過程で，このＯＭを通過するメッセージ

表2　TMの構造

要素	内容
Object Pool	ロックを獲得したオブジェクトの情報
Probe Queue	OMと同じ役目を持つ．

TMは，OMに対しTからの要求か，もしくはマネージャ間のロック情報の転送を行なう．OMは，TMを介した

Tからの要求に対して処理を行なったり，TMに対しマネージャ間のロック情報を転送したりする．データオブジェクトは，本システムではOMに含まれるものと解釈する．よってsuperC^{2}による本設計は，TとTM，そしてOMの3つのモジュールによって構成されるものとする．

表1，2に，TMとOMの内部で管理されるデータ構造について説明する．

4.3 設計結果と考察

本システムの設計結果を表1に示す．

表3 本システムの設計結果

object name	method	message
transaction	5	2
transaction_manager	7	4
dataobject_manager	6	5

表3において，" transaction "は，データオブジェクトに対して，ロックもしくはロックの解除要求を送るオブジェクトであり，" transaction_manager "は，オブジェクト " transaction "を管理するマネージャ・オブジェクトである．また，" dataobject_manager "は，データオブジェクトを管理するマネージャ・オブジェクトである．" transaction_manager "と" dataobject_manger "間には，マネージャ間のロック情報が転送される．methodは，オブジェクト内で定義されたメソッド数を表し，messageは，他のオブジェクトへ転送するメッセージ・パッシング数を表している．この設計結果による考察を，以下に示す．

(1) 逐次処理，並行処理を問わず，メッセージ通信方式は統一した記述方式によって，シンプルなプログラム構造が実現され，プログラム自体も簡素化することが可能である．

(2) プロセス間通信がメッセージ通信方式に隠蔽されているため，ネットワークやプロセスを特に意識することなく設計することが可能である．

(3) superC^{2}のメッセージ通信方式は，従来の手続き言語と連続性がないため，オブジェクトの設計方法を良く理解しなければ，誤ったプログラミングをしてしまう可能性がある．ここに，上記の点とのトレードオフが発生している．この通信方式の制限が，ユーザへの負担ともなる可能性がある．

(4) 現在のプロトタイプ版では，メッセージ通信の際，送信できるメッセージの個数とサイズ等に制限があり，設計の妨げとなった．今後，分散システムの記述能力の拡張のためには，この制限の除去もしくは緩和が必要である．

(5) superC^{2}は実行されるオブジェクトとそれが存在するノードとの対応表が必要であるが，現在の版では，これをユーザレベルで作成しなければならない．今後，この処理に要する時間の短縮化および各ノードの特性を生かすため，最適化分散方式を検討する必要がある．

5 おわりに

本論文では，並行オブジェクト指向言語superC^{2}の設計構想とシステムの概要について述べた．また，分散システムに発生するリソース・デッドロックを，検出して解消するためのアルゴリズムをsuperC^{2}の適用例として挙げ，実用的な面からの評価を行なった．

superC^{2}は，並列処理で問題となる並行動作部分の抽出といった複雑な問題や，ネットワークやプロセッサなどのハードウェアを意識せずに，無理のない効果的な分散システムの構築が期待できる．

今後の課題としては，現在superC^{2}が持つ実装上の制限の除去，デバッグ環境の整備，そして最適化分散方式の検討を行っていく予定である．

参考文献

[1] 勝山 他：" 通信ソフトウェア向けオブジェクト指向言語superC "，情報処理学会論文誌，Vol.30, No.2, 1989.

[2] 今井 他：" オブジェクト指向言語superCの並行動作機能の拡張 - 言語仕様とプリコンパイラ "，電子情報通信学会春季全国大会，D-94, 1991.

[3] 佐藤 他：" オブジェクト指向言語superCの並行動作機能の拡張 - 動作環境 - "，電子情報通信学会春季全国大会，D-95, 1991.

[4] 米澤 明憲：" 並列オブジェクト指向言語ABCL/1による並列処理記述とその枠組の研究 "，電子情報通信学会論文誌，Vol.J71-D, No.8, pp.1415-1422, 1988.

[5] R.E. フィルマン 他：" 強調型計算システム "，マグロウヒルブック (1986).

[6] 吉田 他：" 視覚化支援機能をもつ並行プログラミング言語CABLEとその処理系の実現 "，電子情報通信学会技術報告，Vol.89, No.166, CPSY89-22, 1989.

[7] Mrina Roesler and Walter A. Burkhard, " Resolution of Deadlocks in Object-Oreinted Distributed Systems ", IEEE Trans.on Computer, Vol.38, No.8, August 1989.