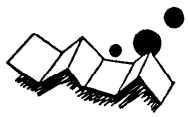
解　説



高信頼化技術とアーキテクチャ†

相　磯　秀　夫††

1. はじめに

システムの高信頼化は，(i)故障の発生を極力おさえる，(ii)仮に故障が発生しても，正常動作に自ら復帰する機能を備えることによって達成される．ここでいう“故障”とは論理回路の欠陥現象からプログラムの誤りまでさまざまなものがあり，いずれも誤動作の原因になるものである．実際に，これらの故障に対して，論理回路レベルからシステム・レベルに至るまで高信頼化技法は種々考案されている．計算機アーキテクチャの観点からも，ハードウェアの物理的故障との関連において，誤り検出・訂正コード，マルチプロセッサ方式，RAS など積極的に取りあげられているが，ソフトウェアや論理的な故障との関連において論じたものは意外に少ない[1]．

ここでは，故障という用語を物理的な欠陥から論理的な誤りまで広義にとらえたとき，信頼度の高いデータ処理やソフトウェア開発の生産性を上げるために役立つ計算機アーキテクチャ・レベルの高信頼化技法について簡単に考察してみたい．

2. アーキテクチャの役割

もともとアーキテクチャとは“プログラマから見たハードウェアの論理仕様”を意味していたが，最近では計算機の論理構造全体あるいはハードウェアとソフトウェアとのインタフェース全体をさすことが多い．要するに，いろいろなレベルのプログラマが直接的あるいは間接的に所望する機能を効率よく，しかも実用的なレベルでサポートするために必須で，基本的な機能を設定することがアーキテクチャの課題といえる．しかしながら，今までの計算機に設定されてきたアーキテクチャ・レベルの機能（たとえば機械命令）はきわめて単純で，しかも固定的であるために，プログラマは解くべき問題から計算機が備える機能への意味変換に大変な苦労を負わされている．その結果，信頼性の高いソフトウェアの開発が難しく，しかも生産性の改善も低い値にとどまっている．

計算機の応用分野が拡大するにつれ，大規模で複雑なソフトウェアの開発が高い信頼性の下で要求される一方，半導体を中心としたハードウェア技術の著しい進歩がそれらの問題解決に期待できるという事実を考えれば，高信頼性ソフトウェアの開発において，広義の意味の“故障”の発生を極力おさえ，故障の検出，故障の自動修正を可能にする機能や機構をアーキテクチャ・レベルで保証するよう考慮すべき時期にきているように思う．この種の問題解決には，現実的には種種の難問の解決が必要と考えられる．特に論理的な欠陥に関する問題を抜本的に解決しようとすれば，計算機アーキテクチャ・レベルでの高信頼化技法の研究が不可避であろう．

3. 高信頼化の課題

ソフトウェアとの関連において，計算機アーキテクチャに課せられた重要な課題の一つは，

（1）解くべき問題とプログラミング言語

（2）プログラミング言語と機械語

との間に存在する大きなセマンティック・ギャップをいかに縮小するかであるといわれている[1]．

結局，このギャップが論理的な故障をつくり出す可能性を内蔵した信頼性の低いソフトウェアの開発しかできない原因になっている．現在の計算機の下で，論理的な故障の発生を避けようとすれば，プログラマは大変な苦労をしなければならないし，人間としての能力の限界も出てくる．

一例として，開発したプログラムの実行において，計算誤差のような“故障”の発生を防止し，仮に発生しても故障の発見を容易に行い，故障に対して自から適応して誤りを修正することができれば，信頼性の高いデータ処理が期待できる．そのためには，高信頼化

† Fault-Tolerant Computing and Computer Architecture Hideo AISO (Department of Electrical Engineering, Faculty of Science and Technology, Keio University).

†† 慶応義塾大学理工学部電気工学科

の機能・機構をアーキテクチャ・レベルでどこまでもてるかが重要な課題になる．このことは計算機アーキテクチャ・レベルにおけるフォールトトレラント技術あるいは論理的故障に対する一種の適応計算機構を研究することにつながる．

一般に，故障の検出ならびに訂正を含む故障回避はフォールトトレラント機能の第一段階であり，重要なものであるが，ソフトウェアとの関連において，アーキテクチャの観点から有効な主要概念を簡単にまとめれば，

（i） 保護に関するもの：誤り検出・訂正コード，記憶保護機構，ケーパビリティ・アドレス方式，ランタイム・チェック機構など

（ii） データ環境に関するもの：データの自己定義機能，データ構造（オブジェクト）定義及びアクセス機能，データによる命令修飾機能（命令のデータ・タイプからの独立）など

（iii） 計算処理に関するもの：10 進演算，可変長データ処理，任意精度計算，有効けた計算機能などであろう．

一方，故障の発生にともなって，適切な処置を自ら行い，修正する適応機能をアーキテクチャ・レベルで実装している例は少なくないが，種々の計算誤差の累積を回避し，任意精度の計算を保証する試みなどには自動適応機能の有用性が指摘されている[2]．

一般に，このような特定な問題に対して適応機能をもつためには，

（a） プログラムや計算機の内部を監視するモニタ機構

（b） モニタした状態を解析する機構

（c） 過去の状態を蓄積し，適当な個所まで戻れるバックトラック（後戻り制御）機構

（d） 処理アルゴリズムなどを変更して，新しい処理を行うための再計算機構

（e） 場合によっては，過去の経過をデータベースに蓄積し，上記適応機能を効率よく行う学習機構を備えることが必要になる．

このような故障に適応する機能を実現するためには，ダイナミック・マイクロプログラミング方式や人工知能の分野で見られる新しい問題解決手法が有用なことはいうまでもない[3]．

4. おわりに

ここでは，故障という用語を広義の意味にとらえたとき，フォールトトレラント技術と計算機アーキテクチャの関係を，特に信頼度の高いデータ処理を行うためのソフトウェア開発との関連において，簡単に考察した．当然のことながら，計算機アーキテクチャの変更，あるいは新しい機能の追加はソフトウェアという莫大な資産を考えるとそう容易に実現するものではない．蓄積されたソフトウェア資産は有効に利用すべきであり，また有効利用が計算機アーキテクチャ改革の大前提になることはいうまでもない．このような困難があるにもかかわらず，ソフトウェア危機の到来ということを考えれば，アーキテクチャの改革を考慮しなければならない時期にきている．また，本質的な問題解決を図ろうとすれば，ソフトウェアにおける種々の“故障”に対して適応する機能を計算機アーキテクチャ・レベルで考えるべきであろう．

参 考 文 献

1) Myers, G. J.: Advances in Computer Architecture John Wiley (1978).
2) Sakamura, K. et al.: An Automatic Recovery Mechanism to Prevent the Occurrence of Subtract Errors, Proc. 6th Annual Symposium on Computer Architecture (1979).
3) 相磯秀夫他編：ダイナミック・アーキテクチャ，bit 臨時増刊 (1980).

（昭和 57 年 2 月 4 日受付）