高信頼 UNIX「風雅」の試作と評価*

6M－6

土屋芳浩†, 吉岡孝司‡
(株) 富士通研究所§
{tsuchiya, yoshioka}@flab.fujitsu.co.jp

1　はじめに

我々はフォールト・トレランス技術の OS 自身への適用方式を研究し，高い信頼性を持つ OS を実現することができた [2]．我々は，試作した高信頼 OS を実際に動かして，高信頼性を確認するとともに，高信頼化する以前のオリジナルのシステムとの比較を行なつた．
本稿では，我々が開発した高信頼 UNIX「風雅」の試作と評価について報告する．

2　プロトタイプ評価システムの構成

我々は Chorus マイクロカーネルと UNIX サブシステムを「風雅」の実装に用いた [1]．実験のターゲットにしたのは，図 1 に示したように，ethernet 上の PC 三台からなるクラスタで，三台で一つの UNIX のイメージをユーザに与える．ここで三台の名前を図のように，それぞれ key site, slave 1, slave 2 と呼ぶ．なお，使用した PC は，i486/33MHz, 36MB メモリ，IDE disk, Adaptec ISA SCSA HBA である．ユーザのプロセスは，key site にあり，SCSI ディスクのファイル操作についてのシステムコールはすべて，slave 1 にあるファイルマネジャ (FM) により処理される．

我々は，オリジナルの UNIX に対し，高信頼機能を追加する作業を行なった．　FT 化ファイルマネジャ (FTFM) は slave 1 と slave 2 に二重化されており，一つの SCSI ディスクにアクセスできる (図 2)．従って共有ディスク上のユーザのデータは slave 1 の故障時に slave 2 に引き継がれることが保証される．試作システムでは FM 以外のサーバが故障するとシステムダウンになるので FM 以外は key site に配置した．このため key site の故障は考えないことにする．

*Highly Available UNIX "FUGA" Prototype and Evaluation
†Yoshihiro Tsuchiya
‡Takashi Yoshioka
§Fujitsu Laboratories Ltd.

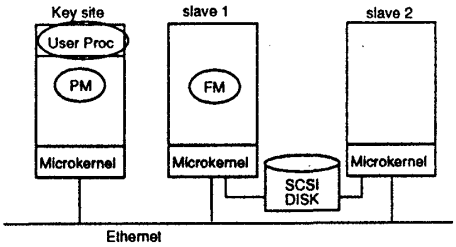

図 1: オリジナルのシステム構成

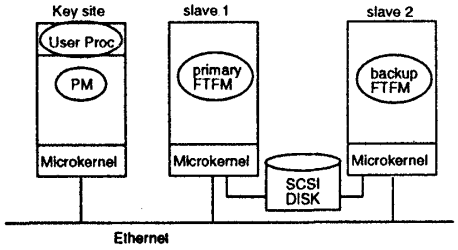

図 2: 風雅の構成 (primary: 現用, backup: 待機)

3　耐故障性の確認実験

試作システムの耐故障性を確認する為に次のような実験を行った．(1) ハードウェアの故障をエミュレートする為，slave の電源を動作中に落す．(2) マイクロカーネルのソフト故障をエミュレートする為に，動作中に slave のリセットボタンを押す．(3) 故障発生ツールを使つて，OS の内部で人工的なソフト故障を起す．この結果，システムコールの実行中であっても，エラーはユーザには見えず，引き継ぎ時間の分だけ処理が遅れたように見えるだけであった．また，slave 1, slave 2 を交互に落しても期待される動作をした．

4　測定結果と評価

4.1　コードの変更量

システムの高信頼化の手間を測る客観的な量として，オリジナルのコードに対する変更量を測定した．ソースコードは C 言語及び C++ で書かれている．システム全体及び FM のみの，オリジナルに対する変更量の割合

表 1: コード変更量の割合 (%)

	追加	削除	新規
全体	0.5	0.02	8.0
FTFM	0.8	0.05	4.5

表 2: 性能比較 −1

	オリジナル	風雅
context switch(msec)	1.3	1.4
system call(usec)	33	35

(行数) は, 以下の通りである (表 1).

全体の変更量は 10% を切り, その大部分が新規追加のコードであることがわかる. また, 既存のサーバである FM に対する変更も少い. 「風雅」の高信頼化はオリジナルのシステムの論理に対する変更はほとんど必要なく, 比較的容易であった. そのことが量的にも反映されていると言える.

4.2 性能比較

高信頼化のオーバーヘッドの見積りの為に, オリジナルとの性能比較を行なつた. 高信頼化による変更とは直接関係ないと思われる, context switch と, system call で若干の性能の劣化が見られる (表 2). 使用しているメモリ領域が大きくなっていることが性能に効いている可能性がある.

また, 通常実行時の性能を知るために行なった, SDET と Kenbus [3] (いずれも single user) の実行結果 (経過時間) を示す (表 3). Kenbus においては両者ほぼ同じであるが, SDET では, オリジナルの性能の 50% 強であることがわかる. オーバーヘッド 178 sec(i.e. 378 − 200) のうち, 30 sec 程度が PA によるオーバーヘッドであり, 残りが SA と, SA により発生する通信のオーバーヘッドであることがわかった.

表 3: 性能比較 −2

	オリジナル	風雅
SDET(sec)	200	378
Kenbus(scripts/hour)	13.84	13.81

PA, SA のいずれも, 今回の実装は最適なものではないので, 今後さらにオリジナルの性能に近づけることが可能であると考えている. 例えば, SA が同期するデータ量は現在の現在の半分にできると見積もられている. PA における論理の最適化と含めて最終的には SDET で 300 sec 以下 (オリジナルの性能の 7 割程度) になるものと思われる.

4.3 故障救済時間

故障の救済にかかる時間を測定した. 実験の方法は, 現実の利用状況を反映するために, SDET ベンチマーク動作中に, slave 1 の電源を落し, ハードウェア故障をエミュレートするものである. 故障救済時間は, 故障検出から通常処理再開までと定義した. その結果, 故障救済に 10 秒から 13 秒程度必要であることが分った. また, 引き継ぎの際に, ファイルシステムの持つデータの引き渡し方を工夫することでさらに救済時間を短縮できることがわかった.

5 まとめ

我々の試作した「風雅」は, プロセスペア技術により, UNIX を高信頼化することに成功した. システムのエラーをユーザに隠蔽することも実現している. また, 高信頼化の為のコードの変更作業はオリジナルのシステムの論理の変更をせずに行なうことができることもわかった. 今回の試作ではオリジナルに比べて性能が落ることが問題点であるが, さらに最適化することにより, オリジナルの性能に近づくことができる.

6 参考文献

[1] Batlivala et al., "Experience with SVR4 Over CHORUS", USENIX Workshop on Micro-Kernels and Other Kernel Architectures, April 27, 1992, pp. 223–242.

[2] 岸本他, "高信頼 UNIX 「風雅」", 第 52 回情報処理全国大会論文集 6M1–5, 1996.

[3] SPEC consortium, "SPEC SDM release1.0 benchmarks".