オペレーティング・システム 38 － 1
（１９８８．２．２６）

分散ファイル・システムにおけるオンデマンド・レプリケーションによるファイルアクセス効率の改善について

芦原 評、 清水 謙多郎、 前川 守、 浜野 純
東京大学理学部情報科学科

ファイル配置問題は多様に研究されてきたが、過去の多くの解はシステムの完全な知識と集中制御に基づくもので分散システム上での実現には不適であり、それゆえ極めて特殊な用途か、理論的研究の範囲を出なかった。本論文では、我々がダイナミック・レプリカと名付けた動的・部分的なレプリカファイルとその生成を決定する分散的アルゴリズムを示した。決定は各ノードにより動的、自律的に行われ、局所的判断のみによるにもかかわらず、例によっては過去のアルゴリズムによる最適解にまさる場合さえあることが示された。

ダイナミック・レプリカの概念の定義の後、ダイナミック・レプリカを生成、維持するための体系を検討し、性能改善のための条件を求める。その結果多くの現実的場面においてダイナミックレプリカがシステムの性能を改善することが示された。

"File Access Improvements in Distributed Systems By On-Demand Reprication of Files" (in Japanese)
by Hyo ASHIHARA, Kentaro SHIMIZU, Mamoru MAEKAWA and Jun HAMANO (Department of Information Science, Fuculty of Science, University of Tokyo, 7-3-1 Bunkyo-ku, Tokyo, 153, Japan)

Technical Report 87-26, Department of Information Science, University of Tokyo (in English)

１．はじめに

分散ファイル・システムにおいて、ファイル配置の決定は理論・実際の両面で極めて重要な問題である。

与えられた条件のもとで、最適配置、すなわち、平均応答時間あるいは通信コストを最小にするような、分散システム内の各ノードへのファイルの割り当て（もしくはその近似）を求める必要がある。その配置は、システムの設計時、あるいは個々のファイルの生成時などに外部から指定することもできるが、ユーザーがシステムの物理的構造にとらわれず、すべてのファイルがローカルに存在するかのように扱える（透過的）システムを、効率を犠牲にすることなく構成するためには、システム自身が最適配置を決定できることが望ましい。

このとき、同一ファイルのレプリカ（コピー）を複数のノードに置くことがありうる。レプリカはシステムの信頼性を確保すると同時に、性能の向上のためにも重要である。しかしながら、コピーファイルを許した場合のファイル配置問題はＮＰ困難であることが知られている。

レプリケーションの特殊な形と考えられるものとして、キャッシング、すなわちプロセスによる遠隔ファイルの一時的取込みがある。本論文では、リモートファイルを必要に応じてローカルにコピーする機構（ダイナミック・レプリカと呼ぶ）を提案し、その効果を解析する。ダイナミック・レプリカは、最初ページテーブルのみが存在し、実体のレプリケーションはページごとにオンデマンドで行われる。

１．１　分散システム

分散ファイル・システム、又は分散データベース・システムについての多くの研究の中で、たとえばＳＤＤ－１[1]やＬＯＣＵＳ[22]は信頼性、性能の双方を指向したレプリカを扱うものであるが、信頼性・可用性についての面をより強調している。

性能向上を主目的としてもいくつかの方式が提供されており、ＩＴＣ[13]ではリモート・アクセスに際して常にファイル全体をキャッシュするという方式をとっている。また読み出し専用のファイルのレプリケーションも行う。ＣＦＳ[14]も、読み出し専用ファイルへのリモート・アクセスに限ってキャッシングを行う。

このほか分散ファイル・システムに関する最近の試みについてはSvobodova[19]やTanenbaum[20]に概観されているが、ファイルのレプリケーションをサポートしたものは比較的少なく、特にレプリカの生成、存在とその位置に関する透過性を実現したものは筆者の知る限りない。これは、レプリケーションに際して何等かの人間的判断が要求されることを意味する。

１．２　レプリケーションと性能

レプリケーションによる性能向上は、原理的には、読み出しの時に最も近いレプリカにアクセスすればよいかわり、書き込みの際にはすべてのレプリカを更新しなければならないとすれば、そのファイルに対する読み出しが更新に比して十分多いとき期待できる。ファイルが読み出し専用の場合は更新に関連した複雑な制御機構が不要なため特に有利である。

その定量的な評価としてはCasey[2]やMuro[10]のものがある。これらは一般のファイル配置問題の一要素としてレプリカの存在を考慮したものである。

Caseyは読み出しが更新より多い場合コスト上レプリケーションが有利となりうるが、完全結合ネットワークの各ノードにおいて読み出しに対する更新の割合が一定ならば、その逆数がオリジナルを除くレプリカの数の限界であることを示す一方、ＡＲＰＡネットワークのデータを使用した例によりこの比が０．４の場合のレプリケーションを含む最適配置を示している。Fisherも同じくＡＲＰＡの例で、更新比率０．４でレプリケーションが有利となる結果を得ている[6]。さらにFisherの結果の一つは、レプリカが少なすぎるよりは多すぎる方が良いことをも示唆している。

一方Muroは乱数を用いて生成したネットワークモデル上の実験により、細かい条件をレプリケーションに有利に単純化した場合でも、最適配置にレプリカが含まれるのは更新比率０．２程度以下の場合に限り、ほとんどの場合多数のレプリカの配置は現実的でないと述べている。しかしながらMuroのモデルは平面上のノードの分布を一様、各ノードからのファイル参照の頻度をすべて等しいと仮定したもので、これが両者の結論の差の主たる原因と思われる。

１．３　ファイル配置

ファイル配置問題についても、Chu[3]以来多くの研究がなされている[5,21]が、考慮すべき要素があまりにも多く、一般性か現実性の一方を犠牲にすることになりがちである。複雑な条件をすべて組み入れた場合、ケース・スタディの形をとらざるを得ない。しかも理論的にはきわめて単純なモデルでさえＮＰ困難であるため、現実のシステムにおけるファイル配置決定のため、さまざまな「実用的」アルゴリズムが考案されている。この際、レプリケーションは特例的であって現実の有効性は低く、問題を徒らに複雑化するとして、考慮から外されることも多い。

たとえば山崎はレプリケーションを認めなくともファイル相互の関係によるコストの変化を考えた場合同じくＮＰ困難であるとしている[25]。

これら過去に行われた研究はネットワークをグローバルに見ての議論が多い。ネットワーク全域にわたるデータの交換と集中的な配置決定は困難であり、実用に供するファイル・アロケーション決定のアルゴリズムは効率的かつ分散向きでなければならない。これらはネットワーク・トポロジや各ノードのファイルへのアクセス頻度を既知の固定したものと見ているが、実際には、ハードウェアによる制限など、さまざまな「現実的」条件を考慮しつつ、ダイナミックな変動に対処しなければならない。そして、ファイル配置をそのように変更する場合、再配置自体に要するコストも考慮しなくてはならない。単一のサービスのためでなく、公共、グループ、個人を含むさまざまなユーザーのためのさまざまな目的を持つ大システムの場合これらは顕著であって、Muroの仮定とは逆に、ノードごとのアクセス頻度に大きな偏りがありうる場合が重要である。レプリケーションは、アクセス頻度が「多極集中」型の場合に効果を発揮する（これは、ネットワーク中２つのノードだけが実際にファイルにアクセスする場合を考えれば、容易にわかる）。特にほとんどのファイルが（結果的に）ローカルなアクセスしかされない（しかし無視できない例外は常に存在する）ようなシステムにおいても、アクセスの効率と配置決定の効率を両立させねばならない。

他方一般に資源管理・負荷分散と呼ばれる問題も概念上同じ目的を持つが、逆に往々にして議論がプロセス単位に限定され、また与えられた資源と要求をマッチさせるという形を取るためレプリケーションは無視されがちである。Kuroseはレプリケーションを認めないか、たかだかレプリカの数があらかじめ与えられた場合を考察している[8]。

また、１つのファイルの部分によっても条件の偏りがあり、性能上レプリケーションを考えるとき常にファイル全体をレプリケートすることは無駄、時には有害になる。

１．４　ダイナミック・レプリケーション

本論文においては、リモートファイルに対するアクセスが必然的にページ転送を伴うタイプのシステムを対象に、ダイナミックにページ単位で生成、消滅するレプリカ（ダイナミック・レプリカと呼ぶ）を考えることにより、この問題の一つの実用的な解決を計る。

すなわちダイナミック・レプリカ（以下ＤＲ）は生成時には構造のみが存在し、実体はページ単位でアクセス要求があるご

とに転送されてレプリカにとどまり、一定時間アクセスがなければ消滅する。　これはキャッシュの拡張と見ることもできるが、キャッシュと異なりプロセス間で共有することができる。（図１）

各ノードの立場に立った場合、ファイルへのリモートアクセスはレプリカ生成の要求であると考え、その点にレプリカを作ることが作らない場合より有利かどうかを判断するという形になる。

この方式は現在開発中の分散オペレーティング・システムＧＡＬＡＸＹ[9]において採用したものである。

その有効性の議論が本稿の目的である。任意のファイル、ノードにおいて、ＤＲを作るか否かの０－１問題として、単純化されたアクセスモデルに基づき通信コストの評価を行い、いくつかの場合についてその可否を論ずる。

２．モデル

ネットワーク上のノードに、Ｐページのオリジナルファイル Ｆが存在するとする。オリジナルの位置を移動させることはないとし、以後、このノードとＦを同一視する。このＦを、距離Ｌ（＝ページあたりの伝送コスト、一定とする）の位置Ｄ付近にあるノードまたはノード群Ｄ’のプロセスからアクセスする。このとき、Ｄにレプリカを作ることによりコストが改善されるかどうかを検討する。

ＤとＤ’の距離はいずれもＬ’であり、原則としてＬ’《Ｌである。そのためＤ’とＦの距離はＬとする。（図１）

各プロセスはキャッシュあるいはバッファを持ち得るが、それらはプロセスの一部とみなす。外部から観測されるのは、プロセスからのページ要求、すなわちプロセスが必要とする（読み出し、書き込みを問わず）ファイルのページがキャッシュになかった場合の要求であり、これが単位時間にｆの割合で起こる。またファイルへのページの書き戻しがｗの割合で起こる。

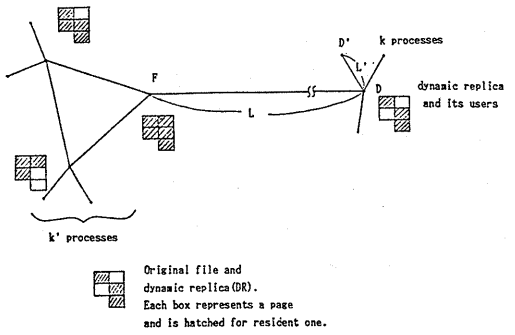

図１　ダイナミック・レプリカの概念図

箱はファイルと各ページを、斜線部はそのうちＤＲに実在するものを示す

２．１　変数

以下の変数を定義する。

Ｌ　：ＦとＤ、およびＦとＤ’の距離
Ｌ’：ＤとＤ’の距離　（Ｌ’＜＜Ｌ）
ｆ　：単位時間中のページ要求
ｗ　：単位時間中の書き込み要求
ｔ　：時間
ｋ　：Ｄ’においてファイルにアクセスするプロセスの数
ｋ’：同じファイルにアクセスする上記以外のプロセスの数
Ｐ　：　ファイルのページ数
ｖ(ｐ)：ページ要求において、第ｐページが要求される確率
ｖｗ(ｐ)：書き込みにおいて、第ｐページに書き込まれる確率
ａ(ｐ，ｔ)：ページｐが時刻ｔまでにアクセスされる確率
Ｓ(ｔ)：時刻ｔにおいてＤＲに実在するページ数

２．２　レプリカのない場合

ｋ個のプロセスがそれぞれ時間あたりｆの比率でＦからページを読み出し、ｗの比率で書き戻す。ｔ時間の間の総通信コストは

$$C_1(t) = k(f+w)tL$$

となる。

２．３　レプリカのある場合

次にＤにＤＲを置く。(ＤとＤＲを同一視する)これを使用するプロセス(群)Ｄ’からの距離はすべてＬ’とする。Ｄ＝Ｄ’ならＬ’＝０としてよい。Ｄに実在しないページが要求されるとＦから読み込みを行い、そのままＤにとどまる。(図１)

ページｐが確率ｖ(ｐ)でランダムにアクセスされると仮定する。

以後はｐを連続変数とみなすと

$$\int_0^P v(p)dp = 1 \quad (0 \leqq p \leqq P)$$

一度アクセスされたページはＤにとどまるので、ファイルの第ｐページが時刻ｔまでにアクセスされている確率

$$a(p, t) = 1 - EXP(-v(p)kft)$$

より、
時刻ｔにおいてＤに存在するページの総計は

$$S(t) = \int_0^P a(p, t)dp$$

$$S(t) \leqq kft \quad (t \geqq 0)$$

は直観的にも明らか。

［例］

１）
ｖ(ｐ)＝１／Ｐ(すなわち一様)と仮定すれば

$$S(t) = P(1 - EXP(-fkt/P))$$

この場合を以後一様アクセスと呼ぶ。固定したｔに対し一様アクセスのときＳ(ｔ)は最大となる。

2)　　2／P　　0≦p≦P／2
v(p)= {
　　0　　P／2＜p≦P

（全ページの半分だけが実際にアクセスされる）

S(t)=(P／2)(1－EXP(－2fkt／P))

3)
v(p)=(2／P²)p　（ページ数に比例）

S(t)=P－(P²／2fkt)(1－EXP(－2fkt／P))

(図2)

通信コストは

$$C_2(t)=S(t)(L+L')+kwt(L+L')+(fkt-S(t))L'.$$

L'＝0(ローカル)の場合を考えると

C₃(t)=S(t)L+kwtL
　　≦C₁

ファイルのアクセス特性に関する研究[11,12,15,18,23,24]は、ここでのいくつかの仮定はやや単純化がすぎることを示唆しているが、我々の目的には十分な近似を与える。以下の議論において、一般性を保持するため、t→∞での長期的条件と、一様アクセスなどの代表的ケースを論ずるにとどめ、関数S(t)の正確な形に依存することは極力避けたが、個々のファイルの特性が予め知られていればtの小さな状況ではより現実的な判定が可能になる。SheltzerはOSのネーム・サービスに限ったアクセス特性とキャッシングの効率を論じている[16]。

一般にv(p)は未知であり、f、wはファイル全体にわたる平均であって、ページごとの相当する値の変動を含む。実際には、ファイルがクラスタに分割され、同一ファイル内で性質の異なる部分に分かれる場合、クラスタごとにレプリケーションを行う。(すなわち、クラスタ単位で構造が、ページ単位で実体が生成される)クラスタのサイズについては、実現上のオーバーヘッドとの兼ね合いになる。部分的に読み出し専用が指定されている(ρ＝∞)場合は特に重要である。逆に、特に書き込みの多いことがわかっているページにレプリケーションを禁止するフラグを立てるなども考えられる。

v(p)に偏りがあると、部分的にレプリケーションが不利なページが現われうるが、全体としては一様のときよりよくなることに注意する。

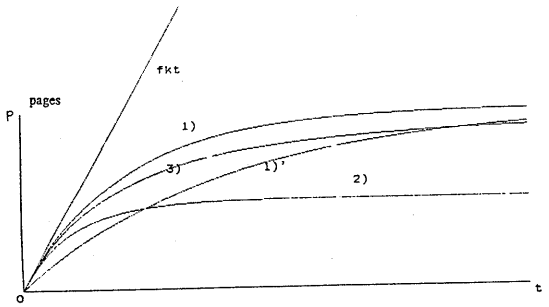

図2　[例]の3種類のアクセス特性によるS(t)の変化

1)'はkfが半分の場合の一様アクセスを示す

2．4　更新

次にFが更新される場合を考える。
F側にも同様のプロセスがk'あるとする。(他のレプリカの使用者も含む)
それらのプロセスの更新アクセスが第pページに落ちる確率をvw(p)とすれば、D上に当該ページが存在するときそれを書き換える通信コストは

(転送されるページ数)
=(全更新の頻度)×
　∫∫(時刻tにページpが更新される確率)
　　×(tにpがDに存在する確率)dtdp
として

$$C_4(t)=k'w\int_0^P\int_0^t v_w(p)a(p,t)\,dt\,dp\;L$$

$$=k'wt-\frac{k'w}{kf}\int_0^P\frac{v_w(p)}{v(p)}(1-e^{-v(p)kft})dp\;L$$

=(k'w／kf)(fkt－S*(t))L　とおく。

S*(t)=∫(vw(p)／v(p))a(p,t)dp

vw(p)=v(p)と仮定すれば
S*(t)=S(t)

一様アクセスの場合
vwによらず、同じく
S*(t)=S(t)

(∵∫vwdp=1)

これは一般には保証されないが、最悪の場合でもS*(t)＞0であるから、この性質が以下の議論において致命的になることはない。

2．5　利得

ここで、DRの有効性について考察する。
コストの差は、

$$\Delta_1=C_1-(C_2+C_4)$$

$$=\left(\frac{k'w}{kf}S^*(t)-S(t)\right)L-\left\{wL'+fL'+\left(\frac{k'w}{kf}\right)fL-fL\right\}\cdot kt$$

k'／k＝κ、f／w＝ρ、L'／L＝λとおくと、第二項は

－(λ＋ρλ＋κ－ρ)kwtL

ここから,

(1＋κ)＜(1＋ρ)(1－λ)　……(1)
または
(w／(w＋f))＜(k／(k＋k'))((L－L'))

の時、t→∞でレプリカ有利になる。(長期条件)
特に、α＝κ／ρとおけば、λ＝0(L'＝0)では

α＜1

これらのパラメータ中、直感的に最も理解しにくいのはρであるが、これは更新比率の逆数で、ファイルの性質によって決まる定数と見てよい。ただし、機構上書き込みの際にもキャッシングが行われることから、全アクセスに対する更新比率に近い意味を持ち、ρ＞１が期待されるが、保証はされない。
さらにα＝κ／ρはＤの読み出しに対するＦの書き換えの比率と見ることができるから、上の結果は極めて当然のことである。

２．６　遷移

ｔが小さくＳ(ｔ)が卓越する場面について、いくつかの特別な場合を論ずる。これらは現実にもしばしば重要である。

λ＝０、Ｓ*(ｔ)≧Ｓ(ｔ)では、κ／ρ＜１のときｔによらず
$\Delta_1 > 0$。

(１)が満たされるとして、ｔ＝ｔ₀において$\Delta_1 = 0$とおくと、

Ｓ*(ｔ)＝Ｓ(ｔ)の場合
Ｔ₀＝(ρ＋１)λ／(ρ－κ)

(Ｔ(ｔ)＝１－Ｓ(ｔ)／ｋｆｔ、Ｔ₀＝Ｔ(ｔ₀))
０＜Ｔ＜１、ｔ→∞でＴ→１

また時刻ｔにおいてＤＲの存在が有利である条件は
(１＋κ)＜(１＋ρ)(１－λ／Ｔ)　……(１ｔ)

Ｓ*(ｔ)＝０の場合
(１＋κ／Ｔ)＜(１＋ρ)(１－λ／Ｔ)　……(１ｔ’)

逆にＳ*＞Ｓであれば(１ｔ)より良くなる。

ρ＝∞(ｗ＝０)であれば、(１)は常に成立し、Ｓ*によらず

Ｔ₀＝λ

[例]
一様アクセスでは
λ＝１－Ｐ(１－EXP(－ｆｋｔ₀／Ｐ))／ｆｋｔ₀

Ｔ(Ｐ／ｋｆ)≒０．３７であるから、λ＜０．３７であれば、Ｐページが伝送されるまでにはＤＲが有利になっている。
他のｖ(ｐ)においてはこれより良くなる。

κ＝０(ｋ’＝０)では、同様に

Ｔ₀＝(１＋１／ρ)λ

Ｆ側にプロセスがない(κ＝０)の場合、安全上の配慮を別とすれば、ＤＲの更新をただちにＦに書き戻す必要はない。このケースでは、最終的に書き戻すべきページ数
Ｓｗ(ｔ)＝∫(１－EXP(－ｖｗｋｗｔ))ｄｐとおくと
Ｓｗ(ｔ)≦ｋｗｔ

Δ_1＋ｋｗｔＬ－Ｓｗ(ｔ)

$$= -S(t)L - S_w(t)L - (wL' + fL' - fL - wL)kt$$

$$= -S(t)L - S_w(t)L + (1+\rho)(1-\lambda)kwLt$$

長期条件は常に成立し、ｔとの関係もρ＝∞の場合とほぼ同じになる。

２．７　拡張

ここまで、Ｄ側にｋ個、Ｆ側にｋ’個のプロセスが独立・並列に走っていると見てきた。あるノードで全く同じプロセスが同時にあるいは繰り返し使われる場合、アクセスされるページはほとんど同じであり、理想的には更新分を除いて新たな転送は不要になると考えられる。
別な見方として、すべてのプロセスについてはｖ(ｐ)＝０→ｖｗ(ｐ)＝０であるが、Ｄ側とＦ側のプロセスが違えばこの制限は必要でない。
これらを念頭に、
Ｑ＝ｌｉｍ(ｔ→∞)∫ｖｗ(ｐ)ａ(ｐ，ｔ)ｄｐ
すなわちｖ(ｐ)＞０となるｐについてのｖｗ(ｐ)の積分
(０≦Ｑ≦１)とおく。

ＤＲの飽和以後、
Δ_2＝ｋｆｔＬ－Ｑｋ’ｗｔＬ－ｋｗｔＬ’－ｆｋｔＬ’
＝－(λ＋ρλ＋Ｑκ－ρ)ｋｗｔＬ

κをＱκにおきかえたものとなる。
条件は、
(１＋Ｑκ)＜(１＋ρ)(１－λ)　……(２)

であって、当然(１)より有利である。

次に、ハードウェアの制限その他により、レプリカの容量を制限した場合を考える。
現実にはリプレースメント・アルゴリズム等を考える必要が生ずる。

１)Ｓ*(ｔ)＝Ｓ(ｔ)の場合

ｔ＝ｔ₁以降Ｄに実在するページ数が増えないとすると、以後のアクセスに関して

$$\Delta_3 = (d\Delta_1 / dt)\big|_{t=t_1} \cdot t$$

$$= \left[\left(\frac{k'w}{kf} - 1 \right) S'(t_1)L - \left\{ wL' + fL' + \left(\frac{k'w}{kf} - 1 \right) fL \right\} k \right] \cdot t$$

＝ＲｆｋｔＬ－ｗｋｔＬ’－ｆｋｔＬ’－Ｒｗｋ’ｔＬ
＝(Ｒρ－Ｒκ－λ－ρλ)ｋｗｔＬ

(Ｒ≡１－Ｓ’(ｔ₁)／ｋｆ＝∫ｖ(ｐ)ａ(ｐ，ｔ₁)ｄｐ、
０＜Ｒ≦１、単調増加)

(一様アクセスではＲ＝Ｓ(ｔ₁)／Ｐ)

$\Delta_3 > 0$の条件は

(１＋κ)＜(１＋ρ)(１－λ／Ｒ)……(３)

すなわち(１)に対しλをλ／Ｒで置き換えた結果となる。λ＜Ｒである必要があるのをはじめ一般には生成の条件・生成後の効率共に不利になるが、λ＝０の場合は条件には影響を受けない。

２)一般の場合
Ｓ(ｔ)≠Ｓ*(ｔ)でありうるから、

Δ_3＝(Ｒρ－Ｒ*κ－λ－ρλ)ｋｗｔＬ
条件は
(１＋Ｒ*κ／Ｒ)＜(１＋ρ)(１－λ／Ｒ)　……(３*)

(Ｒ*≡Ｑ－Ｓ*’(ｔ₁)／ｋｆ)

となって、複雑である。
ｖ、ｖｗが未知であるから、あらかじめ対応することは困難であるが、
前出のクラスタリングによって問題をかなり避けることができる。

［例］

$$v(p)=\begin{cases}\frac{2}{3P} & \\ \frac{4}{3P} & \end{cases} \qquad v_w(p)=\begin{cases}\frac{2}{P} & (0\le p<\frac{P}{2}) \\ 0 & (\frac{P}{2}\le p\le P),\end{cases}$$

なる場合

$$S(t)=P\left(1-\frac{e^{\frac{-2kft}{3P}}}{2}-\frac{e^{\frac{-4kft}{3P}}}{2}\right)$$

$$S^*(t)=\frac{3}{2}P\left(1-e^{\frac{-2kft}{3P}}\right)$$

（Ｔ≡ｆｋｔ）

$1-R=EXP(-2T/3P)/3 \quad +2EXP(-4T/3P)/3$
$1-R^*=EXP(-2T/3P)$
$\equiv X$

$\therefore R=1-X/3-2X^2/3$
$R^*=1-X$

$\lambda=0$であれば
（３＊）：
$$\alpha=\frac{\kappa}{\rho}<\frac{R}{R^*}=\frac{2X}{3}+1$$
すなわち、$1<\alpha<5/3$のとき、全体ではレプリケーション不可だが、容量を
$X=3(\alpha-1)/2$
$S(t)/P=1-X/2-X^2/2$
$=-(9\alpha^2-12\alpha-5)/8$
までに制限した場合は有利になる。最善となるのは
$X=(3\alpha-1)/4$
$S/P=-(9\alpha^2+6\alpha-35)/32$
のとき。

$\alpha=4/3$ならば容量約１／３の時最善、５／８以上では不利になる。

定性的には、ｖの比較的小さいｐにおいてｖｗ／ｖが大きいと、このような逆転現象が生ずる。逆に、ｖの大きい時ｖｗ／ｖが大きいと、容量の制限は(３)よりさらに不利となる。

２．８　ページの無効化

以上の方式では、インプリメントを考えた場合、オリジナル側が、ＤＲの位置と、存在するページに関する情報を把握しておかねばならないという欠点がある。
信頼性を目的とする「通常の」レプリカの場合、原則として全レプリカを同時に書き換えて整合性を保つが、ＤＲはキャッシュに近い「軽い」存在であるから必ずしも同じ方法を執る必要はない。しかしこの場合でも更新同士の衝突を避けるほかプロセス間通信などによる干渉に対し矛盾が生じないようにしなければならない。
Ｆで書き換えられたページが無効化される場合を一種の理想的状態として考えると、同一ページに続けて書き込みが行われた時に無駄な転送をしなくてよい。またオリジナル側がＤＲを無視できるという特長があり、ページごとのオンデマンド転送という概念との整合性もよい。

キャッシュに対しても同種の機構は必要であるが、性能上ここでは問題としない。（なおこれはいわゆるコンカレンシ・コントロールとは別の問題である）[7]
各ページが時刻ｔにＤに存在する確率ａ(ｔ)について

$$\frac{da}{dt}=(1-a)kfv-ak'wv_w.$$

（$v_w=v$と仮定）　$=kfv-(kf+k'w)va$
これを解いて

$$a(t)=\frac{kf}{kf+k'w}(1-e^{-(kf+k'w)vt})$$

ページがアクセスされた回数の期待値をｂとおくと

$$b=\int_0^t(1-a)vkf\,dt$$
$$=\frac{kk'fwv}{kf+k'w}t+(\frac{kf}{kf+k'w})^2(1-e^{-(kf+k'w)vt})$$

時刻ｔまでにアクセスされるページの総和

$$S_2(t)=\int_0^p b\,dp$$
$$=\frac{kk'fw}{kf+k'w}t+\left(\frac{kf}{kf+k'w}\right)^2 S(\frac{kf+k'w}{kf}t)$$

［例］
一様アクセスの時

$$S_2(t)=\frac{kk'fw}{kf+k'w}t+\left(\frac{kf}{kf+k'w}\right)^2 P(1-e^{-\frac{kf+k'w}{P}t})$$

$$C_6(t)=S_2(t)L+kftL'+k\,wt(L+L')$$
$$=\frac{kk'fw}{kf+k'w}tL+\left(\frac{kf}{kf+k'w}\right)^2 S\left(\frac{kf+k'w}{kf}t\right)L+kftL'+k\,wt(L+L')$$

$$\Delta_4=C_1-C_6$$
$$=kft(L-L')-S_2(t)L-kwtLL'$$
$$=-\left(\frac{\rho}{\rho+\kappa}\right)^2 S\left(\frac{\rho+\kappa}{\rho}t\right)L+\left(\rho-\frac{\kappa\rho}{\rho+\kappa}-\rho\lambda-\lambda\right)kwtL$$

より、$t\to\infty$で$\Delta_4>0$となるのは

$(1+\rho)(\kappa+\rho)\lambda<\rho^2$　……(４)

特に$\lambda=0$であればκ、ρ、ｔによらず常にレプリカの存在が有利になる。

Ｄ側の書き込みの際もＦのみ更新してＤを無効化する(ライトスルー)方式も考えられる。
定常状態で

$$\Delta_4(\kappa=1)+kwL'-\frac{kk'fw}{kf+k'w}tL>0$$

$(1+\rho)(\kappa+\rho)\lambda<\rho^2-\kappa$　……(５)

(４)より悪いが、$\kappa>1$では(１)より良くなる。

以上の二方式を同時に使用すると、
$\Delta_4 + kwL' > 0$において$\kappa \rightarrow \kappa + 1$と置き換えれば
$(1+\rho+\kappa)\lambda < \rho$
あるいは
$(1+\kappa) < (1+\rho+\kappa)(1-\lambda)$ ……(6)
であって、(1)、(4)より有利になる。

(4)、(5)、(6)の条件にとって$v = v_w$が最悪の仮定であることは容易にわかる。ライトスルー方式では、書き込みの多いページが無効化されることから、一般にはさらに有効である。

3．応用

機構としてのレプリケーションを論じた以上の結果は、そのまま実際のシステムにおいて個々のレプリカの生成を決定するアルゴリズムにつながる。κ、ρなどのパラメーターについては、システムがある程度自動的にデータを収集することができる。過去のデータを近い将来のアクセス頻度の予測として用いることにより、レプリカ生成の可否を決定することができる。また逆に使用されていないレプリカは削除する。さらに長期にわたる使用の「実績」によってＤＲと通常のレプリカの間の昇・降格を行うことも考えられる。これによって、「見えざる手」式に要求の変動に自動的に対応することができる。

この議論の結果形成されるファイル配置がグローバルな意味で最適に近いとは限らないが、二点間のみの問題として各ノードが自律的に考えることができるため、分散システム上での意志決定は単純で効率的である。他のレプリカやネットワークのグローバルな構造についてノードが知る必要はない。(オリジナル点に関する情報に含まれる)。逆に言えばノードごとのローカルな条件を直ちに反映できる。ローカルに生成され、ローカルに使用されるようなファイルについてははじめから問題としないですむ。ただしシステムを理解しているユーザーが明示的に指定することを妨げるものではない。

プロセス終了後のレプリカの寿命の設定は将来の研究の対象である。これは非分散システムでのページング、キャッシング[4,17]、またファイルマイグレーションとも類似の問題を含む。

もちろん、より深い分析と、きめ細かいデータによれば、それだけ効果的な判定ができるはずだが、それは個々のシステムの特性にも大きく依存したものとなろう。

実現時にまず問題となるのは、実際には仮定の場合ほどはっきりリモートグループが分離できないことである。　そのため、まず$\lambda = 0$の場合に限って具体的な方法の例を示す。ノードＲにおいて、ある程度以上遠方のファイルＦを呼び出す必要が初めて生じたとき、Ｒに小さなファイルｄを作り、以後のＦに対するアクセスの総数(ｆｋｔ相当)を記録する。一方Ｆには、すべてのノードからの読み出し、書き込みそれぞれの要求の総和が記録される。その、他のノードからの全要求に対する自己の要求が２．において求めた基準を満たせばその次のオープン要求の際Ｒにｄにかわりダイナミック・レプリカＤを生成する。

ネットワーク内の近い距離にあるいくつかのノードがグループ(物理的には、たとえばＬＡＮ)をなしている場合、グループ全体として上記の手続きを行うこともできる。その際、こうして一旦生成されたＤＲをグループ外から使用することを許すか、またそれをレプリカ維持のためのデータにどう算入するか、などの問題が生ずる。慎重に行わないと、状況が変化した後もＤＲが惰性的に存続して不利益を生ずる可能性がある。より大きなλに対しても適用可能な実現法の開発が求められる。

4．　例

すでに述べたように、本方式では、”それ以前より改善される”ことだけが保証され、最適配置にどの程度近づくかはわからない。この点を検証するため、Caseyの用いたネットワークのデータ(詳細は[2]を参照)により試算を行う。(1)式の判定で、ｑｕｅｒｙ／ｕｐｄａｔｅ比をそのままρに使用。ページごとの効果は含まれていない。

4．1　5ノード

(図3)は、５つのノードからなるネットワークの各ノード間の距離及び各ノードからのファイル参照頻度を示している。ここで、ファイルを単一のノードに配置するならノード４または５が最善であり、その時のコストは９１５である。

レプリカを認めた場合の最適配置は(１，４，５)で、コストは７０５。

上のアルゴリズムを各ノードについて適用すると、すべてに条件が成立する。従って、
ノード５にオリジナル、１、２、３、４にＤＲを置くと、コストは７５９。

同じ全ノードにレプリカを置く場合でも、Caseyの示す完全結合のコスト(７７１)と異なる(オリジナルが適した位置に置かれていれば、通常改善される)ことに注意。これは本方式の持つもう一つの利点であって、次の例ではさらに劇的に示される。

4．2　ARPA

更新比率０．１のとき、オリジナルの配置は１０が最適(コスト２．４×10^5)。Caseyによる最適配置は(２，１０，１４)で、コストは１．２×10^5。

２－５、１３－１５、１７－１９をそれぞれグループとしてまとめて計算すると、２－５グループ、１２、１３－１５グループが条件を満たす(更新比率が０．２５以上であれば、この方式ではＤＲは作られない)。２、１２、１４にそれぞれＤＲを生成し、ＤＲはグループ外から使わないとして、コスト１．６×10^5。各ノードが常に最も近いレプリカにアクセスするならば、コストはCaseyの最適解より良い１．１×10^5となる。このことから、複数ノードへのメッセージの最適伝達経路と、それをも考慮した最適配置という新しい問題が導かれる。これについては、他の論文でさらに検討する予定である。

INPUT PARAMETERS　　COST PER MEGABYTE SHIPPED

	QUERY TRAFFIC	UPDATE TRAFFIC	FILE 1	NODES 2	3	4	5
U	1	2	0	6	12	9	6
5	3	24	3	6	0	6	12	9
2	4	24	4	12	6	0	6	12
R	5	24	5	9	12	6	0
S	6	24	6	9	12	6	0

(a) The five-node example

COST PER MEGABYTE SHIPPED

		QUERY TRAFFIC	FILE 1	NODES 2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
U	1	5	0	75	75	75	75	350	450	450	540	900	1050	2700	2700	2700	2700	2610	2610	2610	2610
5	2	31	75	0	5	5	5	300	400	450	500	920	1060	2110	2720	2720	2720	2100	2640	2640	2640
R	3	7	75	5	0	5	5	300	400	450	500	920	1060	2110	2720	2720	2720	2100	2640	2640	2640
2	4	6	75	5	5	0	5	300	400	450	500	920	1060	2110	2720	2720	2720	2100	2640	2640	2640
S	5	10	75	5	5	5	0	300	400	450	500	920	1060	2110	2720	2720	2720	2100	2640	2640	2640
	6	4	350	300	300	300	300	0	150	200	500	710	850	1940	2570	2570	2570	2530	2150	2150	2150
	7	3	450	400	400	400	400	150	0	150	420	810	730	1850	2470	2470	2470	2400	2330	2330	2330
	8	10	490	450	450	450	450	200	150	0	240	430	870	1870	2090	2090	2090	2210	2170	2170	2170
	9	9	540	500	500	500	500	500	420	240	0	280	430	1460	2090	2090	2090	2050	1980	1980	1980
	10	60	900	920	920	920	920	710	810	430	280	0	150	1250	1860	1860	1860	1800	1730	1730	1730
	11	3	1050	1060	1060	1060	1060	850	730	870	430	150	0	1100	1760	1760	1760	1650	1610	1610	1610
	12	20	2080	2110	2110	2110	2110	1940	1850	1870	1460	1250	1100	0	610	620	620	640	600	600	600
	13	20	2700	2720	2720	2720	2720	2570	2470	2090	2090	1860	1760	610	0	40	40	200	340	340	340
	14	21	2700	2720	2720	2720	2720	2570	2470	2090	2090	1860	1760	620	40	0	40	200	340	340	340
	15	3	2700	2720	2720	2720	2720	2570	2470	2090	2090	1860	1760	620	40	40	0	200	320	320	320
	16	5	2670	2700	2700	2700	2700	2530	2400	2210	2050	1800	1650	640	200	200	200	0	80	80	80
	17	4	2610	2640	2640	2640	2640	2450	2330	2170	1980	1730	1610	600	340	320	320	80	0	10	10
	18	5	2610	2640	2640	2640	2640	2450	2330	2170	1980	1730	1610	600	340	320	320	80	10	0	5
	19	7	2610	2640	2640	2640	2640	2450	2330	2170	1980	1730	1610	600	340	320	320	80	10	5	0

(b) The ARPA example

図３　ネットワークの例　（Casey[2]による）

ノード間の通信コストと通信量を示す

５．おわりに

以上で、オリジナル側の更新に際して余分な通信を必要とする１）方式であっても、κに比してρが大きい（特に、$\kappa \ll 1$（ｍａｙｂｅ　＝０）、あるいはｗ《１）場合、ダイナミックレプリケーションが効率を高めることが示された。またレプリカの整合性に関する制約を緩めればさらに効率を上げることができる。

効率を論じるに際し問題となる要素には、通信コストの他、ストレージコスト、計算コスト、反応時間などがある。実際の運用にあたってはこれらを考慮に入れねばならないが、個々のノードにおけるストレージコストやストレージ制限を含めた計算は容易である。

通信時間についても、距離の定義を変更すれば上記の結果はほぼ妥当する。ただし、通信コストの場合、リモート／ローカル、ＬＡＮ内／遠距離のアクセスについてλは十分小さいと見てよいが、時間においてはレプリカを経由することによる遅れは無視できないことが多い。またコンカレンシコントロールに要する時間も大きく、実現時のメカニズム、アルゴリズムに強く依存する。

ファイル配置問題は多様に研究されてきたが、過去の多くの解はシステムの完全な知識と集中制御に基づくもので分散システム上での実現には不適であり、それゆえ極めて特殊な用途か、理論的研究の範囲を出なかった。本論文では、我々がダイナミック・レプリカと名付けた動的・部分的なレプリカファイルとその生成を決定する分散的アルゴリズムを示した。決定は各ノードにより動的、自律的に行われ、局所的判断のみであるにもかかわらず、例によっては過去のアルゴリズムによる最適解にまさる場合さえあることが示された。

謝辞
本論文をまとめるにあたり、熱心な討論と多くの貴重な助言をいただいたＧＡＬＡＸＹ計画のメンバーに深く感謝する。

参考文献

[1] Bernstein, P. A. ,et al. :" The concurrency control mechanism of SDD-1: a system for distributed databases (the fully redundant case)", IEEE Trans. Softw. Eng., Vol.SE-4, No.3, (May 1978).

[2] Casey, R. G. :" Allocation of copies of a file in an information network", in Proc. AFIPS Spring Joint Comput.Conf., (1972).

[3] Chu, W. W. :" Optimal file allocation in a multiple computer system" IEEE Trans.on Comput. Vol.C-18, NO.10 (Oct. 1969).

[4] Denning,P.J.:" Virtual memory" Comput. Surv., Vol.2 No.3 (Sep. 1970).

[5] Dowdy,L.W., D.V.Foster:" Comparative models of the file assignment problem" ,Comput. Surv. Vol.14, No.2 (June 1982),

[6] Fisher,M.L., et al.:" Database location in computer networks" JACM, Vol.27, No.4, (Oct. 1980).

[7] Hamano,J:Master's Thesis,To appear.

[8] Kurose, J.F. :" A microeconomic approach to optimal file allocation" in Proc. of the 6th Intern. Conf.on Distrib.Comput.Sys.,IEEE, (May 1986).

[9] Maekawa, M, et al.:" Object Management and Address Space of GALAXY Holonic Processing System", Tech. Rep. 86 15, Dept. of Information Science, Univ. of Tokyo (Oct. 1986)

[10] Muro,S.,et al.:" Evaluation of the file redundancy in distributed database systems",IEEE Trans.Softw. Eng., Vol. SE-11 No.2, (Feb. 1985).

[11] Ousterhout,J.K.,et al.:" A Trace-Driven Analysis of the UNIX 4.2 BSD File System",in Proc.of the 10th Symp. on Oper. Sys. Principles.(1985).

[12] Rodoriguez-Rosell, J. :" Empirical Data Reference Behavior in Data Base Systems" IEEE Computer (Nov. 1987)

[13] Satyanarayanan,M.,et al.:" The ITC Distributed File System:Principles and Design" in Proc.of the 10th Symp. on Oper. Sys. Principles,(1985).

[14] Schroeder,M.D., et al.:" A Caching File System For a Programmer's Workstation" in Proc.of the 10th Symp.on Oper. Sys. Principles,(1985).

[15] Shedler,G.S., C.Tung:" Locality in Page Reference Strings" SIAM J.Comput., Vol.1, No.3 (Sept. 1972).

[16] Sheltzer,A.B., et al.:" Name Service Locality and Cache Design in a Distributed Operating System" in Proc. of the 6th Intern. Conf.on Distrib.Comp.Sys.,IEEE,(May 1986).

[17] Smith,A.J.:" Analysis of Long Term File Reference Patterns for Application to File Migration Algorithms" , IEEE Trans. on Softw. Eng., Vol.SE-7, No.4,(July 1981).

[18] Smith,A.J.:" Cache Memories", Comput.Surv.,Vol.14, No.3,(Sep.1982).

[19] Svobodova ,L.:" File Servers for Network-based Distributed Systems", Comput. Surv., Vol.16, No.4, (Dec. 1984).

[20] Tanenbaum, A.S., and R.van Renesse:" Distributed Operationg Systems", Comput. Surv., Vol.17, No.4, (Dec.1985).

[21] Wah,B.W.:" File Placement on Distributed Computer Systems", IEEE Computer, (Jan. 1984).

[22] Walker,B.,et al.:" The LOCUS Distributed Operation System", ACM SIGOPS Oper. Sys.rev. 17,5,(Oct.1983).

[23] 畑下　豊仁　他：ＵＮＩＸワークステーションにおけるディスク・アクセス特性とディスク・キャッシュの考察：情報処理学会論文誌　Vol.28, No.6, (June 1987).

[24] 菅　隆志　他：ディスク・キャッシュ装置のシミュレーションによる効果測定：情報処理学会論文誌　Vol.22, No.1, (Jan.1981).

[25] 山崎　浩　他：分散型データベースシステムにおけるファイル配置問題の計算の複雑さについて　電子情報通信学会論文誌　D　Vol.J70-D No.4, (Apr. 1987).