Chord プロトコルを活用したシステム開発の実際

大阪市立大学大学院
創造都市研究科
藤田昭人

まずは自己紹介から・・・

- 大阪市立大学大学院　創造都市研究科の学生です（D1）
- 本業はフリーランスのプログラマです（23年目）
- 京都産業大学の非常勤講師もやっています（プログラミングの授業）
- 専門はオペレーティング・システムです（UNIXカーネル屋）
  - 仮想記憶(VM)が好きです
  - 仕事の関係で TCP/IP の拡張や分散システムの開発もやっていました

- P2P と関わるようになったのは永続アーカイブ (Persistent Archive) がキッカケでした
  - これは本来、図書館情報学でのテーマです
  - **「インターネット上に流通する全てのデジタル・コンテンツを数百年オーダーで長期保存する方法は・・・」**といったお題です。
  - 工学の人間にはほとんど笑い話に聞こえますが真剣に悩んでいる人もいるらしい・・・

- 修士時代その筋の同期生に「今のコンピュータでは無理!!」と言い切られたのにカチンと来たので「じゃぁ、やってやろうじゃねぇか!!!!」と大見得を切ってしまったのが事の始まりです。

私にとってP2Pとは・・・（1）

- 博士課程での研究テーマです
- 「永続アーカイブの問題を美しく解決する唯一の鍵？」と考えています
- できれば卒業後の商売の種になるとうれしいなぁと思ってます

- 永続アーカイブの問題については修士時代にヒアリングをしました
  - 某大規模図書館でのヒアリングでの僕の突っ込み
    - 大規模クラスターやデータ・グリッドを使うのはいいけど分散化アプローチが完全分割 (Strict Partitioning) だけというのはかっちょ悪くねぇ？
    - そもそもディスクを買うお金がない場合はどうするの？
  - 担当者の答えは「諦めるしかないですねぇ・・・」でした。

- 個人的には釈然としない気分・・・

私にとってP2Pとは・・・（2）

- 「そんなはずはねぇだろう…」と調べてみたら「おっ!!! やれるんじゃねぇの？」と呼べるものがすぐに見つかりました。**OceanStore** です。
  - *“ OceanStore can be used to create very large digital libraries and repositories for scientific data. Both of these applications require massive quantities of storage, which in turn require complicated management. OceanStore provides a common mechanism for storing and managing these large data collections. It replicates data for durability and availability. Its deep archival storage mechanisms permit information to survive in the face of global disaster. Further, OceanStore benefits these applications by providing for seamless migration of data to where it is needed. For example, OceanStore can quickly disseminate vast streams of data from physics laboratories to the researchers around the world who analyze such data. ”*
    OceanStore は科学データ用の非常に大きなデジタル・ライブラリやリポジトリを作成するために使用することができます。これらのアプリケーションはいずれも大量のストレージを必要とし、さらに複雑な管理を要求します。OceanStore はこのような大きなデータ・コレクションを格納・管理するために共通のメカニズムを提供します。耐久性と可用性のためにデータの複製を行い、そのディープ・アーカイバル・ストレージ・メカニズム (deep archival storage mechanisms) は広域の災害に直面しても情報が残存することを可能にします。さらに OceanStore は、必要に応じてデータが透過的に移動する機能を提供することにより、これらのアプリケーションに役立ちます。例えば OceanStore は巨大なデータ・ストリームを物理学研究所からそのデータを解析する世界中の研究者に速やかに配布することができます。
  - OceanStore: An Architecture for Global-Scale Persisten Storage
    John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao..
    Appears in Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), November 2000
    http://oceanstore.cs.berkeley.edu/publications/papers/pdf/asplos00.pdf
- だから **OceanStore を使えばOK** と考えていたのです。
  ご他聞に漏れず修士学生になりたてのころは実に楽観的だった。
  ですが・・・

OceanStore へのアプローチ（1）

- OceanStore とそのプロトタイプである Pond について詳しく調べてみると・・・
    - Pond: the OceanStore Prototype
      Sean Rhea, Patrick Eaton, Dennis Geels, Hakim Weatherspoon, Ben Zhao, and John Kubiatowicz.
      Appears in Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03), March 2003
      http://oceanstore.cs.berkeley.edu/publications/papers/pdf/fast2003-pond.pdf
- 論文での勇ましさとは裏腹に…
    - Pond の開発は2003年ごろに停止している
    - 残された Java による実装はシミュレータのみ…これが正しく動いているのか全く分からない
    - コアの一部となっている Tapestry の開発担当者はチームから離脱
    - 残るメンバーも Bamboo だとか OpenDHT とか言い出している…
- ぜんぜんダメダメじゃん!!

OceanStore へのアプローチ（2）

- **この段階までで修士課程の期間の 2/3 を使い果たしてしまった!!**
- でも OceanStore の論文が取っ掛かりとなって・・・
  - Structured P2P が永続アーカイブの足場になることだけは確信できた
  - P2P を前提とした分散データ共有インフラストラクチュアという方針も見えてきた
  - P2P の研究事例はまだ他にもあるし・・・
  - 今更、後戻りできるだけの時間の余裕もないし・・・
- このまま P2P で突っ走るしかないでしょう！
- **そうだ、Chord で行こう**　←修士論文提出締切まで残り7ヶ月

Chord とは？（1）

- MIT の Parallel & Distributed Operating Systems Group (PDOS) で開発されました
- DHash との組み合わせで分散ハッシュテーブル (DHT) として機能し、次のような特徴があります。
  - コンシステント・ハッシュを活用して1次元のハッシュ空間にデータとノードをマップする
  - ルックアップの高速化のためにフィンガー・テーブルを活用する
  - 各ノードが自律的・周期的に実行する Stabilization でオーバーレイ構造を維持する
- ルックアップ動作の概要は Tomo さんの「P2Pと分散ハッシュテーブル～その2」を参照してください。

Chord とは？（2）

- Chord は 2001 年頃に発表された一連の構造化オーバーレイ (Structured Overlay) のルックアップ プロトコルの1つで、下記の3つとともに代表的な研究事例とされています
  - Content-Addressable Network (CAN)
    A Scalable Content-Addressable Network
    Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Shenker
    In Proceedings of ACM SIGCOMM 2001
    http://www.icir.org/sylvia/cans.ps
  - Pastry
    Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems
    Antony Rowstron, Peter Druschel
    IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001.
    http://freepastry.org/PAST/pastry.pdf
  - Tapestry
    Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing
    Ben Y. Zhao, John Kubiatowicz, Anthony Joseph
    UCB Tech. Report UCB/CSD-01-1141
    http://www.cs.ucsb.edu/%7Eravenben/publications/CSD-01-1141.pdf
- Chord はこれら4つの研究事例の中でも「もっともシンプルなルックアップ・アルゴリズム」とされています

Chord - 文献（1）

- Chord のアルゴリズムについて解説した同じタイトルの論文が3つあります
  - Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications
    Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan
    ACM SIGCOMM 2001, San Deigo, CA, August 2001, pp. 149-160
    http://pdos.csail.mit.edu/papers/chord:sigcomm01/chord_sigcomm.pdf
    もっとも引用の多い一番有名な論文。

  - Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications
    Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, Hari Balakrishnan
    To Appear in IEEE/ACM Transactions on Networking
    http://pdos.csail.mit.edu/chord/papers/paper-ton.pdf
    上記より2ページ多いバージョン。定理の証明などが追記されている。

  - Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications
    Ion Stoica, Robert Morris, David Liben-Nowell, David Karger, M. Frans Kaashoek, Frank Dabek, Hari Balakrishnan
    Tech. Rep. TR-819, MIT LCS, 2001.
    http://pdos.csail.mit.edu/chord/papers/chord-tn.ps
    上記2本の論文の元になったと思われるフル・バージョン。
    MIT の Technical Report で全45ページの大作。当然、最も詳細。

Chord - 文献（2）

- さらに Chord の活用方法に言及したものとして次のような論文があります
  - Building Peer-to-Peer Systems With Chord, a Distributed Lookup Service
    Frank Dabek, Emma Brunskill, M. Frans Kaashoek, David Karger, Robert Morris, Ion Stoica, Hari Balakrishnan
    Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), May 2001
    http://pdos.csail.mit.edu/papers/chord:hotos01/hotos8.pdf
    初期の開発の途上で書かれたと思われる Chord とその周辺を解説した6ページの論文
    概要を手っ取り早く知るためにはよいかも？
  - Wide-area cooperative storage with CFS
    Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, Ion Stoica
    ACM SOSP 2001, Banff, October 2001
    http://pdos.csail.mit.edu/papers/cfs:sosp01/cfs_sosp.pdf
    Chord の初期のアプリケーションとして有名な Cooperative File System (CFS) の論文
    ファイル配布などを主な目的とした Read-Only の広域分散ファイルシステム

- この他にも MIT では Chord 関連の論文が多数書かれており、
  http://pdos.csail.mit.edu/pubs.html あたりを漁ると、
  うんざりするほど論文が見つけられます
- また、アメリカの大学では分散系の授業の教材として Chord を取り上げているところが多いので、Chord に関連したレポートやスライドなども結構引っかかります

Chord - 実装とその問題点（1）

- Chord は C++ による実装が MIT PDOS から公開されています。
- Chord が動作するのは 主要な UNIX 系プラットホームです。
- Chord 実装は SFS Filesystem Toolkit を使って開発されています。
  したがって予め Self-certifying file system (SFS) をインストールして
  おく必要があります。SFS の参考文献は次のようなものがあります。

  – Separating key management from file system security.
    David Mazieres, Michael Kaminsky, M. Frans Kaashoek, and Emmett Witchel.
    In Proceedings of the 17th ACM Symposium on Operating Systems Principles,
    pages 124-139, Kiawa Island, SC, 1999. ACM.
    http://pdos.csail.mit.edu/papers/sfs:sosp99.pdf
  – A toolkit for user-level file systems.
    David Mazieres
    In Proc. Usenix Technical Conference (June 2001), pp. 261-274.
    http://www.usenix.org/publications/library/proceedings/usenix01/mazieres/mazieres.pdf

- 詳細なコンパイル・インストールの手順に関しては
  Chord HOWTO に記述があります。

Chord - 実装とその問題点（2）

- ここまでの説明だけなら**何の問題もない**と思ってしまうのですが…

- Chord の実装について Chord Project のホームページ (http://pdos.csail.mit.edu/chord/ には 次のようなことが書いてあります

  – At this point no official release for Chord is available, but you are welcome to check out the latest version from the CVS repository. **This version is experimental, under active development, and may frequently be broken.** The Chord HOWTO describes how to download and compile the software.

- つまり文字通りの“AS-IS”というわけで・・・Chord のインストールを試みた人間は**必ずハマります!!**

Chord - 実装とその問題点（3）

- 私が確認したものだけでも以下の問題がありました。

  – SFS のバージョン問題
  Chord HOWTO には SFS-0.7.2 に対応していると記述されていますが、Chord 実装はある時点（詳細は不明）から SFS-0.8pre なる次期バージョンに対応が切り替わっています。ところが SFS のホームページ (http://www.fs.net/sfswww/) では未だにこの SFS-0.8pre のソースコードを公開していません。それどころかサーバーホストである **www.fs.net** が不定期に（数ヶ月に渡って）停止していることさえありました。

  – gcc のバージョン問題
  Chord HOWTO では gcc のバージョンについて次のように注意書きがされています。
  GCC 2.95 (not 2.96, although 3.x should work, for x < 4)
  つまり「2.95 は動くけど、2.96 はダメ、gcc3 は動くかもしれないけど
  未確認で、gcc4 もダメ」ということ。チラッと見ただけだと「ふーん・・・」思うのですが、
  今時 gcc 2.95 を標準バンドルしたディストリビューションなどないので
  実はこの環境を作るのが大変だったりします。

Chord - 実装とその問題点（4）

- さらに・・・
  - autoconf/automake/libtool のバージョン問題
    さらに Makefile.xx などをちゃんとメンテナンスしていないため、かなり古いバージョンの autoconf/automake/libtool を使わないと configure を実行している最中にコケる現象が発生します。
    当然 Makefile は生成されません。
  - コンパイル・エラー問題
    どうにかコンパイルできるようになっても、Anonymous CVS にチェック・インされるソースがコンパイル・チェックされていないため、コンパイル時にコンパイル・エラーが頻発する場合があります。**and may frequently be broken** ということのようです。
  - Chord HOWTO の嘘つき問題
    そしてダメ押しのように… Chord HOWTO 自身もちゃんとメンテナンスされていないので、無事コンパイルが完了しても実行環境の設定でつまずくことがあります。
    つまりチェック・インされるソースと Chord HOWTO の記述の同期が全く取れてない。

Chord - 実装とその問題点（5）

- 結局　**Chord はすばらしいが、MIT PDOS の実装はダメダメ**です。
  - それが証拠に未だに Chord mailing list には最大1～2ヶ月の間隔でインストールの要領の問い合わせがきています。

- 早い話が・・・OceanStore の落とし穴からようやく這い上がったら、どうやら今度は Chord/DHash の落とし穴に落ちた・・・らしい。

Chord 実装 - 取り合えず頑張ってみた（1）

- プラットホームはFreeBSD 4.11 を採用
  - バージョン問題が一番でにくいと思われる
  - 標準コンパイラは gcc 2.95
  - autoconf/automake/libtool は ports に収録されている
    一番古いバージョンを一時的に差し替えて使用

- Anonymous CVS による最新バージョンでのビルド
  - SFS-0.8pre は Debian GNU/Linux のパッケージから拾ってくる
  - コンパイル・エラーが多発したので Chord Mailing List に問い合わせ
    → パッチを送ってもらう
  - パッチを当てたら Chord/DHash のソースはコンパイルできたけど
    CFS のコンパイルはやっぱりダメ。再びメーリングリストで質問したら
    「Chord/DHash のコードは常時アップデートされているが
    CFS のコードは放置されている」ことが判明
    → 断念 （どうなっとるんや **MIT** !!）

Chord 実装 - 取り合えず頑張ってみた(2)

- Anonymous CVS による過去バージョンでのビルド
  - CFS もかつては動いていたであろうという勝手な推測に基づくトライアル
  - Ivy のホームページ の情報を元に 2003/02/20 のソースコードを チェックアウトしてビルド
  - 概ね Chord HOWTO と Ivy HOWTO で紹介されている手順でビルドできた
  - CFS のセットアップは Chord HOWTO の手順どおりで OK
  - Ivy は SFS の認証機構を利用しているため Ivy HOWTO の手順が 理解できずに苦闘する → どうにかする

- CFS/Ivy を実行してみる
  - ファイルシステムをマウントすると数分程度は動く が・・・何かがトリガーになってファイル・アクセスしている プロセスが刺さってしまう
  - 対象ファイルが大きすぎるといきなり刺さる
  - **実用ベースではとても使える代物でない**ことが判明

- 3ヶ月に渡る一連の努力が全て徒労に終わり茫然自失。

Chord 実装 - 取り合えず頑張ってみた(3)

- 更にダメ押しのように・・・
- 指導教官からタイムアウト宣告を受ける
  ←この時点で修士論文提出締切まで残り3ヶ月を切っていた
- 腹立ち紛れに MIXI の分散ハッシュテーブル・コミュで
  「2度と使うか!! MIT PDOS 実装」宣言をする
- 後日談
  - Bamboo の論文(詳細は後述)では Chord の実装を
    動かして比較をしているが、彼らが使ったのは 2003/8/4 の
    CVS snapshot だったらしい。**やっぱりね!!**

オルタナティブ（1）

- 修士論文はホニャララだったけど、とにかく修士課程は修了。
  - 後期課程進学も決まった!!
  - **MIT PDOS 実装の代わりを探さなきゃ!!** ← 今年の3月ごろの話

- MIXI で知ったオルタナティブは…
  - **Overlay Weaver**
    - "オーバレイ構築ツールキットOverlay Weaver"
      首藤一幸, 田中良夫, 関口智嗣:
      情報処理学会論文誌: コンピューティングシステム,
      Vol.47, No.SIG12 (ACS 15), 2006年 9月
      http://www.shudo.net/publications/SACSIS2006-OW/shudo-ACS-15-OverlayWeaver.pdf
      ホームページ: http://overlayweaver.sourceforge.net/index-j.html
    - 首藤さんの Java による実装。（ご本人の解説があったので
      詳細は割愛）MIXI DHT コミュの新年会でも簡単なプレゼンがあって、
      大阪に帰ってから実装をチェック。その時の本音な感想・・・
      なんで3ヶ月前にリリースしてくれへんかったん？
      正直、自分の間の悪さにだいぶん凹んだ。（まぁ、しゃあないんやけどね）

オルタナティブ（2）

- 更に・・・
  - **MACEDON**
    - MACEDON: Methodology for Automatically Creating, Evaluating, and Designing Overlay Networks
      Adolfo Rodriguez, Charles Killian, Sooraj Bhat, Dejan Kostic, and Amin Vahdat
      Proceedings of the USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004), March 2004.
      http://www.cs.ucsd.edu/~vahdat/papers/macedon-nsdi.pdf
      ホームページ: http://macedon.ucsd.edu/
    - 奈良先の門林先生、お勧めの C++ による実装。
      http://macedon.ucsd.edu/release/ からソースコードがダウンロード可能。
      ソースをダウンロードし、マニュアルともどもチェックするが… マニュアルのボリュームにちょっと引く（当然ながら英語やし・・・）
    - **注:** 現在 MACEDONE Project は終結し、
      MACE (http://mace.ucsd.edu/) として開発が続けられている？
      http://mace.ucsd.edu/release/ によれば 0.7.1 (2005/09/19) が最新バージョンみたい。
- 以上の2例は複数の P2P ルーティング・プロトコルをサポートしている

オルタナティブ（3）

- その他には・・・
  - **Internet Indirection Infrastructure (i3)**
    - Internet Indirection Infrastructure
      Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, Sonesh Surana
      Proceedings of ACM SIGCOMM, August, 2002
      ホームページ: http://i3.cs.berkeley.edu/
    - “Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications”の First Author の Ion Stoica が指導していると思われる
      研究プロジェクト。*i3* 自体はアプリケーション・レベルで Multicast/Anycast をサポートするシステム。ルーティング部分に Chord が使用されている。
      Chord Mailing List で**簡易な Chord 実装**として何度か紹介されていた。
  - **Open Chord**
    - ホームページ: http://www.lspi.wiai.uni-bamberg.de/dmsg/software/open_chord/
    - i3 Chord の実装を参考に作られた Java による実装。
      オープンソースのプロジェクトらしい。この実装を使った
      研究報告や論文を今のところ見つけていない
- 他にもいろいろあると思います・・・きっと

オルタナティブ（4）

- で、取り合えず *i3* から手を付けることにしました。
  理由はとにかく **簡易な Chord 実装** だったから・・・
  - 基本的に昔懐かしい**C言語によるデーモン・プログラムの構造**
    - 独立したライブラリとしてビルドされるけど・・・実は内部で子プロセスを folk する
    - 若い学生が UNIX プログラミングを勉強しながら書いたように見える
  - とにかく小さい（ソース全部でも 226KB 程度）
    - 機能も最低限（キー・ベースのルックアップのみ）
    - ソースコードの中身はモジュール化とも抽象化とも無縁 ← 早い話がベタ書き
  - これまでの教訓による学習から・・・
    - **「もし使い物にならなかったとしても犠牲は最低限に抑えられるだろう」**
      ←3たび穴に落ちるのはどうしても避けたかった人
- 指導教官からの**「早いとこプロトタイプを仕上げんかいっ!!!!」**との
  無言(？)の圧力がトリガーになったというか・・・

i3 Chord の実装（1）

- Internet Indirection Infrastructure (i3) の実装
  - 動作概要: http://i3.cs.berkeley.edu/info/how.html
  - ソースコード: http://i3.cs.berkeley.edu/impl/index.html

- *i3* Chord のソースコードをザッと眺めてみると・・・
  "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications" の設計を必ずしも忠実に実装されているわけではない
  - 最大50台までのノードをコンフィグ・ファイルで指定する
  - ノードの接続・離脱は stabilization で行う
  - フィンガーテーブルは successor が先頭、predecessor が最後となる Chord Ring のリスト（**えっ?!**）

- またまたやられたっぽい??　気を取り直して
  コンパイルとセットアップ、実行・・・**ちゃんと動いてるジャン!!**

i3 Chord の実装（2）

- 漠然とした不安を払拭できないまま・・・
- とにかく動作するので作業は続行！！

- そのままでは **読みにくい** and **使いにくい** ので・・・
  ちょっとお色直しをしてみました。

  – *i3* 本体から切り離してスタンドアローン・デーモン化
  – ソケット入出力のみモジュール化 ← メインループの見通しがよくなった
  – アクセス・コントロール機能の削除 ← 実験の邪魔になるだけなので
  – CUI クライアントを書き起こし、その為のソケット入出力を追加

- これでごく普通のネットワーク・デーモンとして動作するようになった
  他にもいろいろ手を入れたくなったけど・・・あと回し

i3 Chord の Lookup（1）

- 本来 Chord はノードをスキップしなら探索をしますが・・・
  i3 Chord は識別子サークルの**線形探索しかしません!!**
  - フィンガー・テーブルは successor から始まるハッシュ値で
    ソートされた単純な双方向リストです。
  - ノードをスキップしながら探索する Chord 特有の
    高速ルックアップは動きません。
  - ルックアップのメッセージ数は *O(log N)* ではなく *O(N)* になります。
- この事実がわかって大分がっかり・・・

i3 Chord の Lookup（2）

- *i3* chord の *find_successor* の動作:
  - コマンド FS のパケットを送信するとノード探索が始まります。
    - 指定された id が自ノードの責任範囲にある場合は
      要求ノードに successor の id を返す。
    - 指定された id が自ノードの責任範囲にない場合は
      predecessor に要求を転送する。
    - 自ノードの successor と predecessor が分からない場合
      （自ノードのフィンガー・テーブルが空の場合）
      要求ノードに自ノードの id を返す。
  - ショートカットのない再帰でノード探索をします。

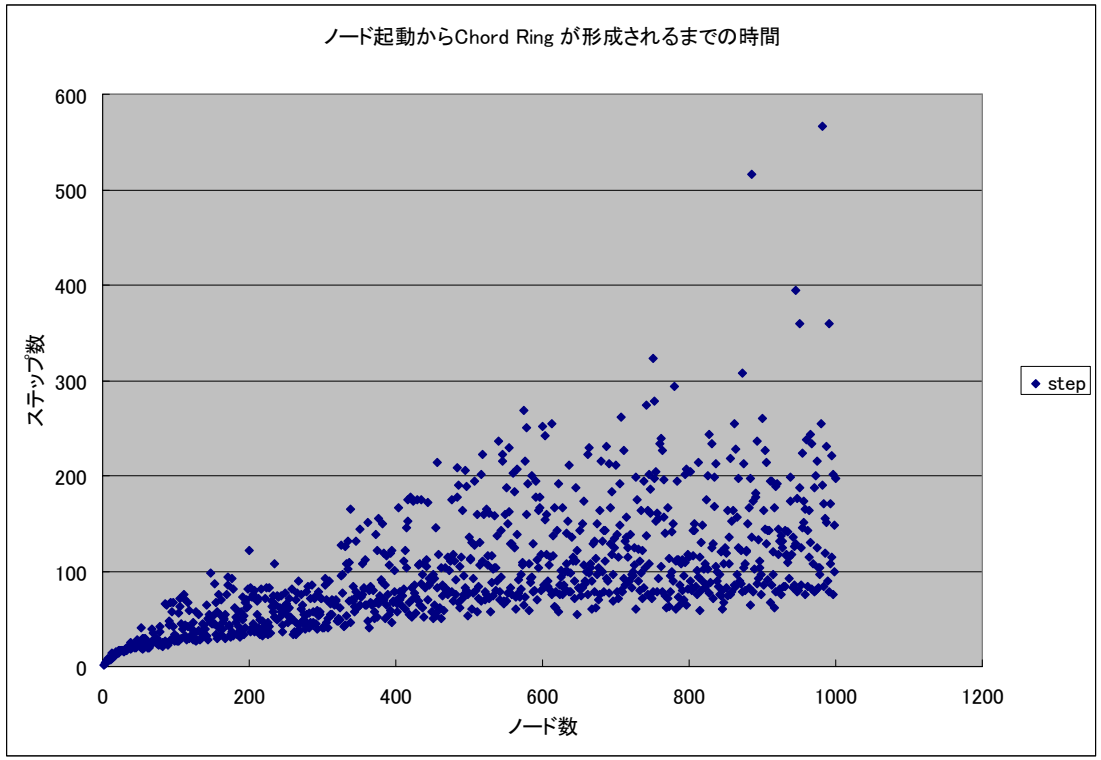
- 探索中に空のフィンガー・テーブルが見つかった場合は コマンド PING/PONG のパケットを送って双方のフィンガー・テーブルに お互いのエントリを追加します。

i3 Chord の Stabilization（1）

- *i3* Chord では 1秒間隔で周期的に次の処理を行っています。
  - フィンガー・テーブルが空の場合は…
    - 全てのノードにコマンド FS とコマンド PING を送信して終了します。
  - フィンガー・テーブルが空でない場合は…
    - コマンド PING に応答しないノードをフィンガー・テーブルから削除します。
    - successor にコマンド STAB を predecessor にコマンド PING を送信して、自ノードの successor/predecessor を安定化させます。
    - フィンガー・テーブルのエントリ総数が NFINGER (=160) を超えないようにエントリを削除します。その際、時計回り方向には NSUCCESSORS (=8)、反時計回り方向には NPREDECESSORS (=3) のエントリは残されます。またエントリの重複も解消します。
    - 既知のノードの中からランダムに選んだ1台のノードに対し、自ノードの successor の間のランダムに選んだ id について *find_successor*（コマンド FS）を実行して正しく実行されるかチェックを行います。
- フィンガー・テーブルと言うよりは Successor リスト？

i3 Chord の Stabilization（2）

- ノード数をパラメータにして起動してから Chord Ring ができあがるまでを計ってみました



- 一連のコード追加作業は概ね３日間、実働８時間程度で終わったので、
やっぱり作業パフォーマンスは良いかも。（僕にとってはね）

さて、困った…（1）

- で、現在の話です。
- 実は今 RDBMS を P2P で共有する研究プロトタイプを作ってます。
  - P2Pを活用した小規模データベースの集約化

    情報処理学会 研究報告 2006-DSM-042
    情報処理学会DSM研究会, Jul.20, 2006
    http://dspace.info.gscc.osaka-cu.ac.jp/~fujita/RESEARCH/DSM042/paper.pdf
- 更にもっと言うと・・・ DHT の上位に乗っかるデータベース制御部分は既に書いてしまっている。
  - SQLite を活用したSQLエンジンを組み込んでる
  - プログラミング言語は**C 言語**で!!

さて、困った…（2）

- いよいよ DHT を組み込む開発の最終段階にある今
  僕が握っている選択肢は次の3つです。
  - 取り合えず *i3* は動いてるんだから、
    それを使ってさっさとプロトタイプを作っちまえ!!!
  - やっぱ Chord のフィンガー・テーブルによる高速ルックアップがないと
    まずいでしょ。*i3* でオリジナルの Chord プロトコルをサポートしないと・・・
  - そもそも C 言語でプログラミングしようとするのが間違い。
    Overlay Weaver を採用してデータベース制御部分を Java で書き直せば？
- う～～～ん、困った!!!　皆さんならどれを選択しますか？
  （横田さんどうです？）

構造化オーバーレイを使った アプリケーションを書くためには（1）

- これまでプロトタイプの設計のためにそれなりに文献を調べたけど・・・
  - 構造化オーバーレイを使ったアプリケーションのプログラミングについて網羅的に解説した文献はまだ見つけられていない。
  - 構造化オーバーレイ・プログラミングの方法論はまだ確立していない？

- その理由について直感的に思いつくことは…
  - 構造化オーバーレイを構成するノードは構造を維持するためにかなり高度な努力をする必要がある。
  - 広域分散システムに共通する一般的な問題（アクセス競合）を解決してくれるとは限らない。
  - ハッシュテーブルの GET/PUT のアナロジーは分かりやすいけど、それだけで実用的な分散アプリケーションを書けるわけではない。

構造化オーバーレイを使った アプリケーションを書くためには（2）

- このあたりが主に実現性や普及という側面で非構造化オーバーレイの アプリケーション（Gnutella とか Winny とか）に水をあけられている理由？ （もちろん他にもいろいろ理由はあるけど）
- 構造化オーバーレイが独立した技術として定着するためには プログラミングの方法論の確立と 開発のための優れた道具が必要なのでは？ →でなければ非構造化オーバーレイアプリケーションにつまみ食いされて終わり

- 僕が構造化オーバーレイに抱く素朴な疑問
  - DHTはただのデータ管理ユーティリティ？ それとも広域分散のためのインフラストラクチュア？
  - 広域分散のためのインフラストラクチュアだとしたら それに相ふさわしいパフォーマンス（特に性能面で）は期待できる？

- これがP2P応用を研究テーマとする僕の博士課程での宿題なのかな？

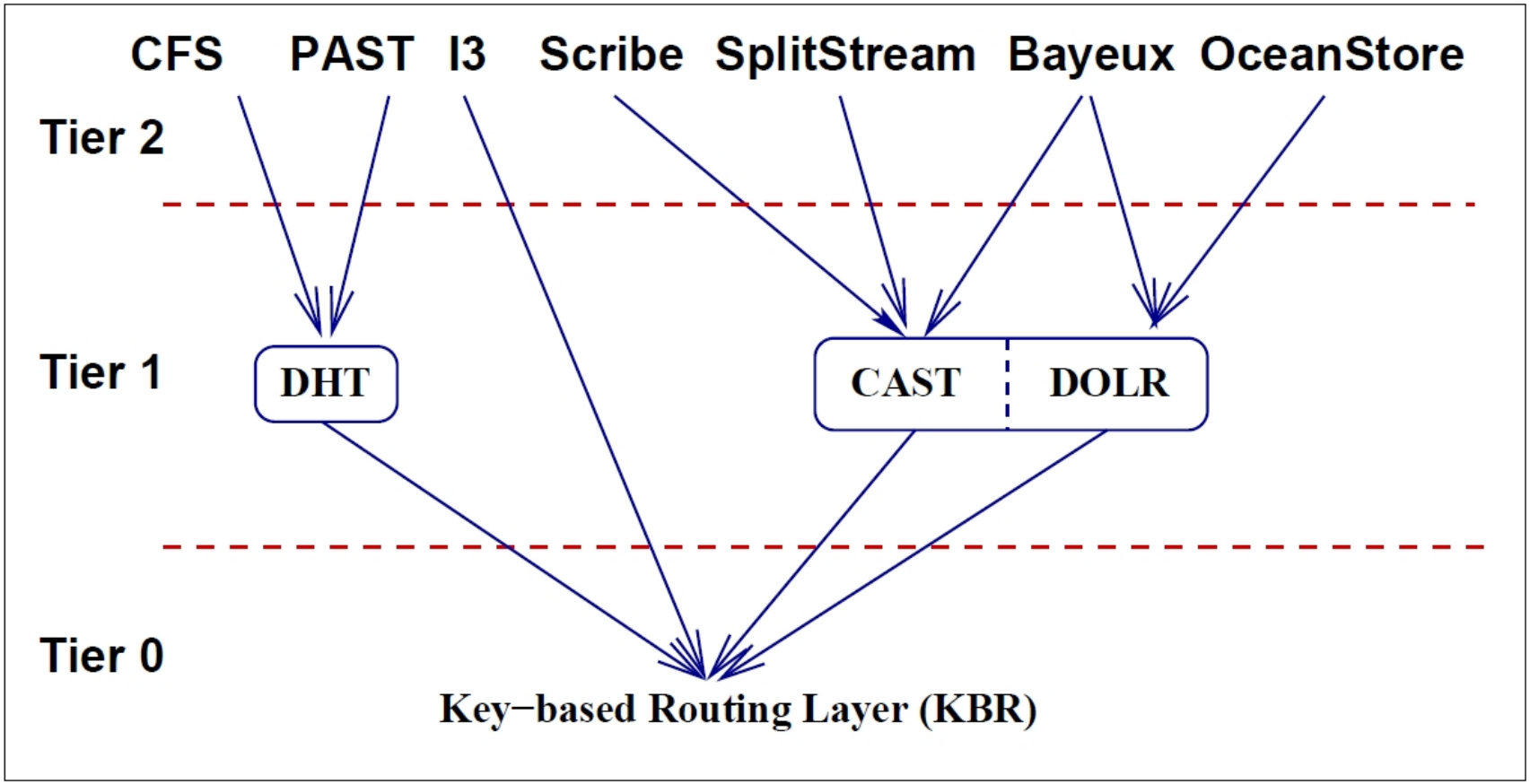
構造化オーバーレイ＝DHT？（1）

- 上位ソフトウェア（特にアプリケーション）から見た場合に
「構造化オーバーレイをどのように理解すればよいのか？」という問題です。
- まず最初に「DHTは構造化オーバーレイを活用する
抽象的概念の1つでしかない」という考え方があります。

  – Towards a Common API for Structured Peer-to-Peer Overlays
  Frank Dabek, Ben Zhao, Peter Druschel, John Kubiatowicz, and Ion Stoica.
  In the Proceedings of the 2nd International Workshop on Peer-to-Peer Systems
  (IPTPS03), Berkeley, CA, February 2003
  http://pdos.csail.mit.edu/papers/iptps:apis/main.pdf

- この論文では最下層のTier 0には KBR、Tier1には DHT、DOLR、CAST
などの抽象的概念、そして Tier2 にはアプリケーションという3階層モデルの
定義を提案しています。

構造化オーバーレイ＝DHT？（2）



構造化オーバーレイ＝DHT？（3）

- 一方「DHTが構造化オーバーレイの基本的な抽象概念で、
  その上にいろいろ積み上げていくべき！」と言う考え方もあります。

  – OpenDHT: A Public DHT Service and Its Uses
    Sean Rhea, Brighten Godfrey, Brad Karp, John Kubiatowicz, Sylvia Ratnasamy,
    Scott Shenker, Ion Stoica, and Harlan Yu.
    Proceedings of ACM SIGCOMM 2005, August 2005.
    http://srhea.net/papers/f230-rhea.pdf

- まずDHTのシンプルな put/get インターフェースを提供し、
  それ以外のことは Recursive Distributed Rendezvous (ReDiR) と
  呼ばれるクライアント・ライブラリで解決することを提案しています。

本当に Chord でいいの？（1）

- Chord に限らず構造化オーバーレイに関わるシステムの
  性能評価は難しいのですけども・・・
  - Analysis of the Evolution of Peer-to-Peer Systems
    David Liben-Nowell, Hari Balakrishnan, and David Karger
    ACM Conf. on Principles of Distributed Computing (PODC),
    Monterey, CA, July 2002.
    http://pdos.csail.mit.edu/chord/papers/podc2002.pdf
- この論文では「システムの状態を維持するためにネットワークに
  参加する必要のあるノードでのレートこそより有益な計測基準である」
  と述べてます。（実は今いちピンと来てないんだけど・・・勉強不足）

本当に Chord でいいの？（2）

- 性能評価の方法についてもう1例・・・
  - Comparing the performance of distributed hash tables under churn
    Jinyang Li, Jeremy Stribling, Thomer M. Gil, Robert Morris, and M. Frans Kaashoek.
    In the Proceedings of the 3rd International Workshop on Peer-to-Peer Systems (IPTPS04), San Diego, CA, February 2004.
    http://pdos.csail.mit.edu/papers/dhtcomparison:iptps04/paper.pdf

  - A performance vs. cost framework for evaluating DHT design tradeoffs under churn
    Jinyang Li, Jeremy Stribling, Robert Morris, M. Frans Kaashoek, and Thomer M. Gil
    In the Proceedings of the 24th Infocom, Miami, FL, March 2005.
    http://pdos.csail.mit.edu/papers/dhtcomparison:infocom05/paper.pdf

- この2つの論文は performance versus cost framework (PVC) という性能評価モデルを提案しています。
- さまざまなルックアップ・アルゴリズムをこのモデルに沿って評価してます
- Chord に関しては「stabilizationがヘコいのは？」と批判しています。
- 前提としてChurn の問題を取り上げている点も重要なのでは？

ルックアップを早くするためには（1）

- Churn ・・・ 直訳だと攪拌 ― オーバーレイの構造が乱れている状態
  - 同時にノードの接続・離脱がたくさん発生するとこういう状態になる
  - 当初は一時的に発生する過渡的な状態だと思ってたけど・・・
  - 構造化オーバーレイでは定常状態として考えたほうがいい？
  - 2003年以降の論文ではよく取り上げられている研究テーマ

- Bamboo
  - Handling Churn in a DHT.
    Sean Rhea, Dennis Geels, Timothy Roscoe, John Kubiatowicz
    Proceedings of the USENIX Annual Technical Conference, June 2004.
    http://srhea.net/papers/bamboo-usenix.pdf

  - Pastry をベースに 開発されたルックアップ・アルゴリズム
  - Churn 状態でもルックアップの性能が落ちない
  - OpenDHT に使われている

ルックアップを早くするためには（2）

- Geometry-Awareness ー ネットワーク的な位置を認識する
  - オーバーレイ上の近傍はアンダーレイでも近いほうがルックアップは早くなる
  - オーバーレイ上の近傍がアンダーレイでは遠方だとネットワーク障害に強くなる

- Vivaldi
  - Vivaldi: A Decentralized Network Coordinate System
    Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris.
    In the Proceedings of the ACM SIGCOMM '04 Conference,
    Portland, Oregon, August 2004
    http://pdos.csail.mit.edu/papers/vivaldi:sigcomm/paper.pdf
  - A Distributed Hash Table.
    Frank Dabek.
    Ph.D. thesis, Massachusetts Institute of Technology, November 2005.
    http://pdos.csail.mit.edu/papers/fdabek-phd-thesis.pdf

  - P2P的に synthetic network coordinate system を生成する
  - 後者の論文では Chord の遅延時間の改善に活用した例を示している

おわりに

- 特にまとめることはありません
- 良い機会なので逆に聞いてみたいんですが・・・

  どうやって構造化オーバーレイの評価実験を行ってますか？
  - お金があるから必要な数だけ実機を調達して・・・
  - お金がないからシミュレーションだけ・・・
  - 少数の実機とシミュレーションの併用（一番ありがち？）
  - PlanetLab に参加することを考えている（本当に問題は解決するの？）

  - その他に画期的な方法があったら教えてください！！