2003－UBI－1 (8)
2003／4／25

ユビキタス環境を情報実体化する組込みデータベース

倉光 君郎
東京大学大学院情報学環
〒113-0033 東京都文京区本郷 7-3-1
kuramitsu@iii.u-tokyo.ac.jp

徳田 英幸
慶應義塾大学政策・メディア研究科
〒252-0816 藤沢市遠藤 5322
hxt@sfc.keio.ac.jp

概要

本稿では、ユビキタスコンピューティング環境における新しい協調分散フレームワークを実現するデータベース技術を応用した情報実体化に関して報告する。

1 はじめに

ユビキタス (ubiquitous) – ラテン語のキリスト (神) の遍在に由来する言葉 – は、次の情報化社会の期待のキーワードとなった。現在では、携帯コンテンツ配信から情報家電まで様々な文脈においてユビキタスが形容されているが、その語意は西洋世界が一神教であったことを思い起こすと興味深い。つまり、ユビキタスとは単に「どこにでも存在する (anywhere)」というだけではなく、それがひとつに統合された結局のところひとつであるということを含んでいる。我々は、このような視点に立って、実世界環境上の小さなコンピュータ・通信機能に対して統合的な情報処理環境を提供することがユビキタスコンピューティングの重要な鍵であると考える。

本稿では、未踏ソフトウェア開発事業を中心に行った「ユビキタス環境を情報空間化するための組込みデータベース」の開発経験を踏まえ、ユビキタスコンピューティング環境 (Ubiquitous Computing Environment: UCE) で知的な協調アプリケーションを構築するための基盤となるデータベース技術を提案していきたい。

我々の提案するアイディアは、極めて単純である。まず、UCE 上のすべてのデバイスに自分自身の情報をプールする小さな組込みデータシステムを搭載する。このデータシステムは、センサーやアクチュエータと連動し、デバイスの状態やコントロール情報をデータベースビューとして提供する。さらに、我々は個々のデータビューを集積し、統合データビューを生成する。この統合ビューは、個々のデバイスの状態をユビキタスコンピューティング環境全体の状態として、ユーザやアプリケーションに提供される。更に特長は、統合ビューを単なる読み取り専用のキャッシュとして提供するのではなく、実体化されたビュー (materialized views) として構築することである。つまり、統合ビューの値を変更すると、それが自動的に各デバイスの状態へ反映されることになる。このようにして、我々は、実体化された UCE のビューを構築することで協調アプリケーションを構築するプラットホームとして提供していく予定である。

実体化ビューの実装やアプリケーションは、リレーショナルデータベース上で盛んに議論されてきた [1]。我々は、ダイナミックで変化に富んだ UCE 上で実体化ビューを実現するため、全く独自の Toibox アーキテクチャを提案する。Toibox アーキテクチャは、半構造データモデル、モバイルデータベース技術などを採用した新しい実体化ビューの管理システムである。

本稿の残りは、次のとおり構成される。第 2 節では、

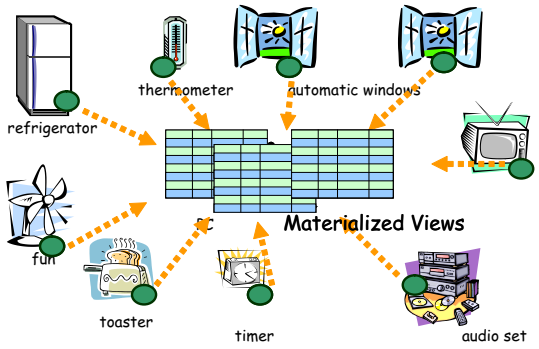

図 1: Toibox におけるデータビューの実体化

ユビキタスコンピューティング環境における情報処理をシナリオベースで議論し、我々の目標を明らかにする。第 3 節では、我々の提案する Toibox アーキテクチャを紹介する。第 4 節では、プロトタイプシステムを報告する。第 5 節では、関連研究と比較し議論する。第 6 節では、本稿を総括し、残された課題を論じる。

2 ユビキタスコンピューティング環境と情報管理の必要性

ユビキタスコンピューティング環境は、刻々と変化する状況にあわせて多種多様なデバイスを協調操作することが求められる。つまり、あらかじめ決め打ち的に用意された相互運用性では対応しきれず、予期せぬ相互運用性 (serendipitous interoperability) が求められる。[2] そのような環境においては、より多くの情報を的確に扱うことが求められる。

我々は、プロジェクトの動機は、まさにユビキタスコンピューティング環境自体をデータベースビューとして捉える点にある。ただし、対象とすべき情報の種類や処理の種類はあまりに多種多様であるため、本節ではサンプルシナリオを用いて、我々の目標と要求を明らかにしていきたい。

2.1 準備

まず、ユビキタスコンピューティング環境 (UCE) について定義しておこう。

定義 2.1 *あるユビキタスコンピューティング環境 (UCE) は、計算能力と通信機能を備えた多数のオブジェクトから構成される。これらのオブジェクトの機能は、多種多様に富んでいる。*

実際に、この節で用いるランニングエグザンプルを考えてみる。

例 2.2 *ある部屋の温度を「快適に」保つことを想定する。この部屋の内側と外側には、CPU と通信機能を搭載した温度計 (センサー) が設けられている。同様に、アクチュエータ機能付の窓もある。もちろん、パソコンやテレビなど他の機器も存在する。これらのデバイスは、すべて CPU が搭載され、お互いに通信可能であると想定する。*

まず「快適な温度」というのが問題となるが、決め打ち的な例として快適な温度を 20 ℃と仮定する。すると、パソコンから温度計の測定結果にあわせて窓を開閉する窓プログラムを書くのは難しくない。例えば、外側の温度が内側の温度に比べ、20 ℃に近ければ窓を開ける、逆なら閉めるという具合である。これは、一見、賢い窓を演出できるかもしれない。

しかし、この窓プログラムが刻々と変化する状況の変化に対応しきれないのは言うまでもない。例えば、新しくエアコンを部屋に取り付けた場合はどうなるだろう？更に、もし台風が近づいて来ていたら、窓を開けてしまっていいのだろうか？決め打ち的なプログラムの限界は、明らかである。

2.2 データビュー：デバイスとロジックの中間層

ここまで読んで読者の方々は、結局のところ、人工知能が必要となるのではないだろうか？と思われたか

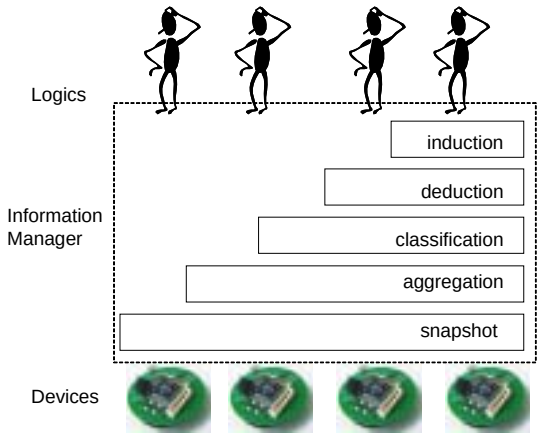

図 2: ユビキタスコンピューティング環境の階層モデル

もしれない。人間と同等に賢い動作を求めるなら、何かしらの推論システムや機械学習のテクニックが必要となるだろう。しかし、我々は、人工知能のテクニックに傾倒する前に、もう少し UCE の情報を扱う技術に注目すべき必要があると考える。

先ほどの例に戻って考えてみよう。ここでは不幸にも外側の温度計が壊れたと仮定してみよう。すると、プログラムは、与えられたコードにしたがってすら、窓の開閉を判断できなくなるだろう。しかし、プログラムが必要なのは、外側の温度計から温度のセンサー値ではなく、外気温のデータである。もし、気象サイトやニュースからそれに相当する情報を得られれば、(正確さの問題はあるにしても) 処理を継続することができる。

ここで我々が注目すべき点は、各デバイス (センサー) が提供する情報とロジック (プログラムやアプリケーション) が必要とする情報の間にはギャップが存在する点である。つまり、デバイスからの情報を集積し、必要に応じて抽象化したり補ったりする機構が必要であり、これは上位ロジックから独立していることが望ましい。我々は、デバイスとロジックの中間層として、情報ビューを構築することを提案する。

2.3 情報ビューの階層構造

我々は、情報ビュー自体も階層的に構成されるべきであると考えている。次は、我々が想定している階層構造である。

スナップショット (snapshot) デバイスから送られてくる生のデータを提供するデータビューである。現在の状態を理解するだけでなく、継続的な監視 (continuous query) も提供する。

集計 (aggregation) 集計とは、外部データベースから必要な情報を補ったり、平均値などの統計処理を行って、データの不足や補う。

抽象化 (classification) より抽象度の高い情報に分類する、もしくは変換する。例えば、部屋にいる人の健康状態から室温が快適か寒いか暑いか分類することが相当する。

ルール (deduction) あらかじめ与えられたルールに従って、データビューを変形させる。例えば、台風が近づいていれば (抽象化のフィルターを通したときは外界の気候が悪くなっていたら)、窓を開ける機能を見えなくする。

学習 (induction) 過去の様々な記録を分析して新しいルールを学習する。例えば、コンピュータをフル稼働すれば部屋の温度が上がるなど。

このように、ユビキタス環境の情報ビューは、低レベルな層ではデータベース技術 (データベース統合、データマイニング) が重要な役割を果たし、より高い層では知識ベースの技法 (知識表現や推論エンジン) が適用できる。我々は、最も基本となるデータベース層をしっかりと構築すれば、上位層において既存のテクニックが適用しやすくなると考える。本稿では、スナップショットや継続的なモニタリング用のデータビューの構築について取り組む。

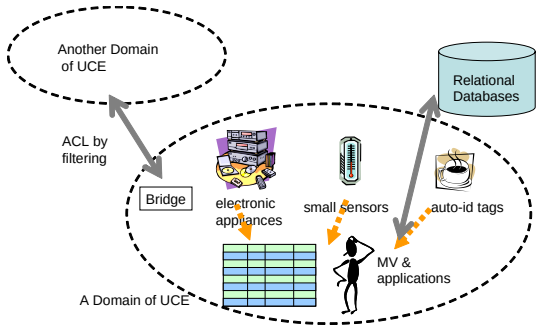

図 3: Toibox のアーキテクチャ

3 Toibox アーキテクチャ

この節では、我々が Toibox と名付けた UCE データベースビュー・アーキテクチャを紹介する。

3.1 概要

Toibox アーキテクチャは、UCE 上のノードを大別して 2 種類に分類し、協調動作環境を構成する。ひとつは、Toi と呼ばれるデータソースを提供するノードである。もうひとつは、Toibox(箱) と呼ばれるデータソースを集積して統合ビューを管理するノードである。図 3 は、概要を示している。

Toi は、基本的に自分自身の情報をプールする小さなデータベースが組み込まれている。各 Toi には、固有のデータベース ID が割り当る。(データベースの詳細は、次小節のデータモデルで述べる。) Toi の特徴は、フットプリントの小さなデバイス上で実装できるように、実際のクエリー処理 (クエリー言語の解析、最適化、実行) を行わず、Toibox データモデルにしたがったデータの入出力を行うだけにした点である。また、データベース ID と Auto-ID [3] などを対応付けることで、非プロセッサデバイスであっても情報アクセス可能なアーキテクチャとなっている。

Toibox は、各 Toi 上のデータベース情報を統合・管理し、ユーザやアプリケーションからのクエリーを処理するためのデータベース機能が搭載されている。各 Toi の状態が変化したら Toibox 上のデータベースも更新され、逆に Toibox 上のデータベースが変更されたら、各 Toi に対し、その変更が反映される。Toi と Toibox 間のインタラクションは、後の「分散モデル」で詳細を述べる。Toibox は、自分自身の情報を管理するため、Toi としての役割を担っても構わない。

現在の Toibox アーキテクチャは、各 Toi や Toibox 上にアクセス制御機構を備えていない。これは、Toibox における UCE の規模を実世界の部屋と想定しているためである。つまり、その部屋へ入室が許可されているなら、各種デバイスへのアクセスが許可されていると想定している。更に、我々は複数の UCEs を結ぶため、専用のブリッジ (birdge) 機能を検討している。Toibox は、ブリッジを通して他の UCEs からデータを得たり、逆にアクセスすることが可能となるが、ブリッジがアクセス制御の役割を果たすことを想定している。ただし、UCE と Toibox アーキテクチャにおけるアクセス制御やプライバシーに関する議論は、開かれた疑問として残っている。

3.2 データモデル

Toibox アーキテクチャは、その組込みデータベースの最も基本となるデータ表現として、半構造データモデルを採用している。[4] 半構造データモデルは、ラベル付き有向グラフで表現され、あらかじめデータベーススキーマを必要とせず、自己記述的にデータ表現を行うことができる。多種多様なデバイスや情報が含まれる UCE を表現するためには、(伝統的なリレーショナルモデルに比べて) 適したモデルといえる。また XML データとの高い親和性もある。

我々は、UCE 上で半構造データモデルを適用するため、いくつか工夫を行った。ひとつは、最も基本的な記述に対して、ラベル (枝) のオントロジー化である。つまり、デバイスは、必ずデータベースルートから device という枝で指向され、更に spec, status, ctrl

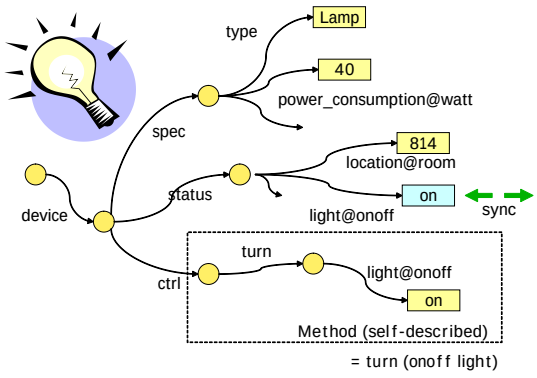

図 4: Toibox のデータモデル

という 3 種類の部分枝を持ち、それぞれ工場出荷時の情報 (不変な情報)、現在の状態を表す情報 (変化する可能性のある情報)、そして機能の情報が記述される。図 4 は、温度計を表している。

第二の改良点は、グラフ上の端点 (葉)、つまりデータ値に相当する部分に対して、リッチなデータタイプを割り当てた点である。(例. 図 4 上の@watt や@onoff など) これによって、実世界で利用される単位を間違えることなく利用することが可能となった。

最後に、デバイスの機能の記述として、従来のようなケイパビリティ (capability) を記述するのではなく、リモート method interface を実体化 (materialized) した点である。具体的には、例えば次の更新クエリーがユーザから発行されたとすると、

```
update $d.ctrl.light@onoff = 'on'.
comit  $d.ctrl
```

これは、Toibox 上のクエリープロセッサでは、次のメソッドをリモート呼び出すことと解釈される。

```
$d.turn('on');
```

各 Toi は、リモート呼び出しによって、状態が変化した場合は、データベース上の状態を更新する必要がある。この場合は、`status.light@onoff = 'on'` となる。このように、データベースの状態を更新するために、メソッドを呼び出すために、安全にデバイスの状態を制御することができる。

3.3 分散モデル

Toibox アーキテクチャでは、各 Toi から Toibox への通信モデルとして、モバイルデータベース技術で用いられる Data Dissemination [5] を採用している。

Toibox システムは、Toi を検出して、そのデータビューを統合し、実体化されたビュー (MV) を構成する機能を備えている。各 MV の更新は、最初に各 Toi のデータビューをバルクコピーするところに始まり、その後は変更された値のみブロードキャストを用いた push 機構で常に反映される。つまり、MV 上の値は、各 Toi 上の最新の値が保持されている。

Toi システム上の実体化されたビュー (MV) 上では、常に各 Toi の最新の値が保持されている。そのため、更新を伴わないクエリー処理は、MV 上だけで処理することが可能である。また、更新を伴うクエリーの場合は、MV 上から更新箇所を探し、該当する Toi に対して、ユニキャストで更新クエリー分だけを転送する。図 5 は、Toibox データモデルに対するクエリー言語の例である。クエリー言語は、SQL 等で馴染みの深い SFW(select-from-where) 型ではなく、XMLQuery 言語で採用された FLWOR(foreach-let-where-operation-result) 型をベースにしてある。このうち、FLWR は Toibox の MV 上で評価され、O だけ各 Toi に転送されて実行される。

4 プロトタイプ実装

我々は、未踏ソフトウェア開発事業において、Toibox アーキテクチャを実現するためのいくつかの初期プロトタイプシステムを開発した。これらのプロトタイプシステムの大半は、Java 言語を用いて開発された。特

```
foreach $o device
  where status.location@floor = 1
      or status.location@floor = 2
{
    update ctrl.trun.light@onoff = 'off' ;
    commit ctrl.turn ;
}
results (
  status.localtion@floor, status.location@room,
  status.light@onoff
);
```

executing on distributed Tois

図 5: Toibox クエリー言語

に、Toi システムは、Personal Java や組込み向けの J2ME 上で動作可能な API を用いて開発してある。

4.1 Toi システム

我々は、3 種類の Toi システムの開発を行った。それぞれの目的は、ネットワークシステムの連動、スモールフットプリントの評価、データベースの記述力である。

分散機能の評価 複数の Toi システムの相互運用を実現するため、我々は SHARP 製 Zaurus SL-B500(Embedded Linux 搭載) を複数台用いて、UCE の実験環境を開発した。Zaurus は、それぞれ無線 LANIEEE801.b で接続され、Personal Java 上で開発された家電エミュレータが搭載されている。これらの家電エミュレータの情報を管理するため、Toi システムがネットワーク上のフロントエンドに置かれている。

小型化の評価 Toi システムは、フットプリントの小さな組込みシステム上で実装できるように設計されている。我々は、C 言語を用いて、Toi システムのコアシステムだけ実装した。そのために要したソースコードは、高々3000 行程度で記述することができた。

記述力の評価 我々は、Toi システムの表現力を評価するため、東芝製 DVD レコーダ RD-XS40 の持つ複雑な状態や機能をデータビュー化を行った。

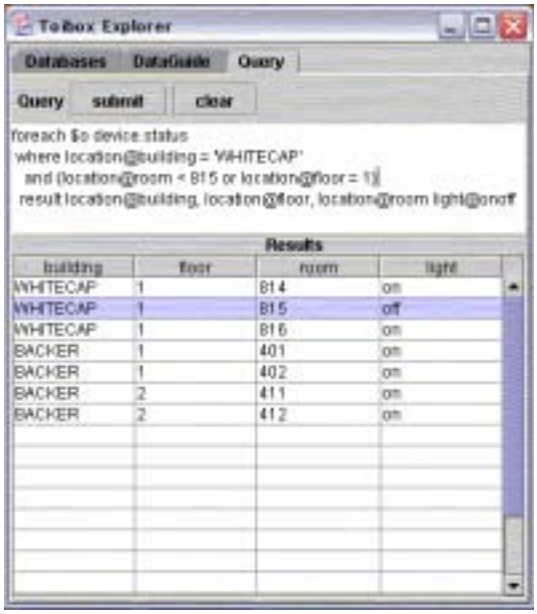

図 6: ToiboxExplorer

4.2 Toibox システム

第 3.3 節で述べた通り、Toi システム上の実体化されたビュー (MV) 上では、常に各 Toi の最新の値が保持されている。そのため、Toi システムは、クエリー処理を MV 上だけで処理することが可能である。図 6 は、ToiboxExplorer と名づけた、Toibox 上の MV に対してクエリー処理を行い、その結果を表示するための GUI ツールである。クエリー結果は、リレーションとして取り出すことができる。

Toibox システムの特徴は、UCE の変化に合わせて刻々と変化するクエリー結果を捉えるため、Continous Query として実行される点である。つまり、クエリー結果に変更が生じた場合は、表示されているテーブル上の値が変更される。

図 7: Toibox の DataGuides ビュー

4.3 データガイド

我々は、Toibox 上でデータベースビューの管理とクエリー操作の支援するため、データガイド (DataGuides) 機能を開発した。データガイドとは、スタンフォード大学で開発された半構造データからスキーマを抽出する技術のひとつである。データガイドを用いれば、UCE 上でどのような種類の情報が提供されているか、個々の Toi を調べる必要なく把握可能になる。図 7 は、開発されたデータガイドシステムの表示例である。データガイドを用いれば、個々のデバイスを意識することなく、UCE 全体への情報アクセスと見ることが可能となる。

5 関連研究

我々は、UCE もしくは実世界環境のコンピューティングを支援する新しいアーキテクチャとして、Toibox を提案してきた。Toibox は、UCE の伝統的なコントロール指向とデータ指向なアプローチに中間に位置付けられる。

コントロール指向のアーキテクチャの代表例は、Sun の Jini や Microsoft PnP に代表される分散オブジェクト指向技術をベースにした方法である。[6, 7] それぞれのデバイスは、仕様や機能を記述したサービスプロファイルをルックアップサーバに登録する。利用するクライアントは、ルックアップサーバから望みのデバイスを探し、各種の機能にアクセス方法である。しかし、ルックアップサーバに登録された記述は、スタティックであり、ダイナミックな変化を反映するものではない。

データ指向の例としては、センサーネットワークとストリームデータベース技術が存在する。[8–10] これは、センサーが捉えた情報を効率よく提供することを考えられており、逆にクエリーとして実世界を更新することは考慮されていない。

Toibox と同じく、コントロール指向とデータ指向の中間的な位置づけと解釈できるのは、Embedded Web サーバ技術の応用である。[11] 近年、多くのネットワーク家電に採用されたシステムで、広く普及した Web ブラウザを通して、情報のアクセスや機器の制御が可能になっている。しかし、Embedded Web サーバは、デバイス間の協調動作を実現することが難しいのが現状である。

6 まとめと今後の課題

今、時代は、個々の家電がネットワークに接続される時代から、それらが複雑にかつ賢く協調動作をすることが期待されるユビキタスコンピューティングの時代へと大きな一歩を踏み出そうとしている。そこでは、ダイナミックに変化する多種多様なユビキタスコンピューティング環境の情報を如何に管理するかが重要となっている。我々は、ユビキタスコンピューティング環境の情報を集積して管理するという視点に基づいて、Toibox アーキテクチャを提案した。Toibox アー

キテクチャは、半構造データモデル、分散データベースと実体化ビュー、更にモバイルデータベースのテクニックを組み合わせた新しいデータベース技術である。

本稿は、未踏ソフトウェア開発事業における成果を中心に Toibox システムを報告した。しかし、我々のプロジェクトは始まったばかりである。データベース技術の精錬だけでなく、セキュリティーや大規模システムへの適用など、多くの取り組むべき課題が残されている。今後は、実験を行いながら研究・開発に取り組んでいく予定である。

謝辞

本研究は、平成１４年度の未踏ソフトウェア開発事業の開発資金の一部を用いて行われた。著者は、本研究プロジェクトを進めるにあたり、積極的に討論され、意義あるご意見を提供してくださった宇田隆哉氏、市村哲氏、伊藤雅仁氏 (東京工科大)、藤田憲正氏、鵜川始陽氏、松村晋吾氏 (京都大)、倉橋誠氏、徳永英治氏、根本将寛氏 (早稲田大)、梅田英和氏 (スカイリー・ネットワークス) に深謝している。

参考文献

[1] Ashish Gupta, and Inderpal Singh Mumick: Maintenance of Materialized Views: Problems, Techniques, and Applications. *Data Engineering Bulletin*, 18(2), pp. 3-18, 1995.

[2] Jeff Heflin, Raphael Volz, and Jonathan Dale, eds. Requirements for a Web Ontology Language. *W3C Working Draft*, 08 July 2002.

[3] Sanjay Sarma, David Brock, and Daniel Engels. Radio Frequency Identification and the Electronic Product Code. *IEEE MICRO*, pp. 50-54, Nov/Dec 2001.

[4] Serge Abiteboul, Peter Buneman, and Dan Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 1999.

[5] Daniel Barbara. Mobile Computing and Databases – A Survey. *IEEE Transaction on Knowledge and Data Engineering*, 11(1), 1999.

[6] Sun Microsystems. Jini Network Technology. Available at `http://wwws.sun.com/software/jini/`.

[7] Universal Plug and Play Forum. Avaiable at http://www.upnp.org/.

[8] Samuel R. Madden, Robert Szewczyk, Michael J. Franklin and David Culler. Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks. Workshop on Mobile Computing and Systems Applications, 2002

[9] Philippe Bonnet, J. E. Gehrke, and Praveen Seshadri. Querying the Physical World. IEEE Personal Communications, 7(5), pp. 10-15, 2000.

[10] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma. Query Processing, Resource Management, and Approximation in a Data Stream Management System. In Proc. of the 2003 Conference on Innovative Data Systems Research (CIDR), January 2003

[11] Hewlett-Packard Company. CoolTown Home. Available at http://cooltown.hp.com/cooltownhome/index.asp.