usp lab.

ユニケージ開発手法
ビッグデータ導入事例

有限会社ユニバーサル・シェル・プログラミング研究所

2013年2月

Universal Shell Programming Laboratory

ユニケージ開発手法によるビッグデータ処理の事例

usp lab.

①バッチ処理の置換え（某クレジット会社）

②複雑なETL（某証券会社）

③複雑なETL（某電力会社）

④大量データ検索（韓国大手検索サイト）

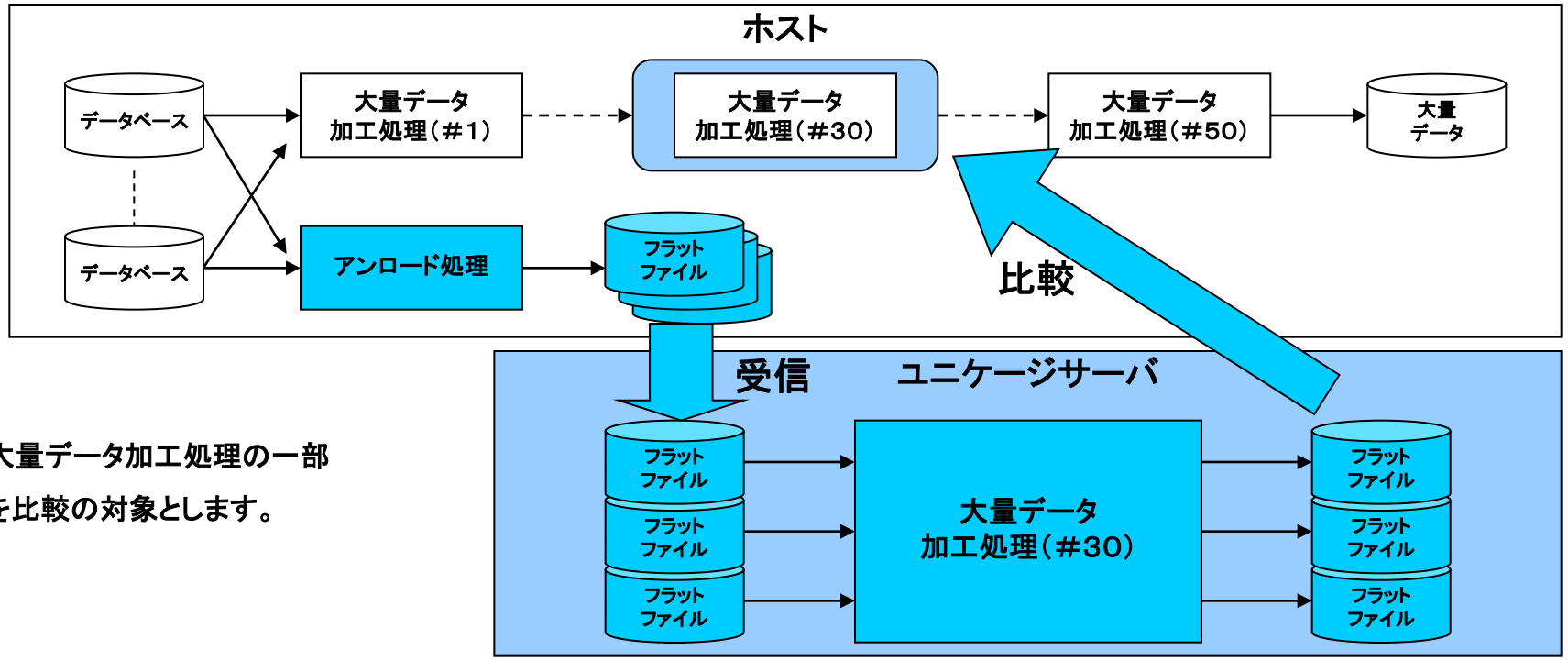
⑤まとめ

Universal Shell Programming Laboratory

①バッチ処理の置換え（某クレジット会社）概要

usp lab.

ホストで、大量データの加工処理を実行しています。
この処理をユニケージへ移行します。
そのため、ホストから処理に必要な元データを受信し、ユニケージで一部の処理を作成し、比較を行っています。



大量データ加工処理の一部
を比較の対象とします。

Universal Shell Programming Laboratory

処理速度

usp lab.

処理時間がCOBOLの**8分の1**になりました。(116.00/929.69=**12.4%**)

ユニケージは、PCサーバ5台(CPU6コアx2、メモリ48GB)での測定結果です。

サーバの台数を増やして分散処理を行えば、さらなる高速化が可能です。

	COBOL	ユニケージ (PCサーバ 1台)	ユニケージ (PCサーバ 5台)
処理時間	929.69分 (15時間29分)	313.58分 (5時間13分)	約116.00分 (1時間56分)
ハードウェア	ホスト (初期費用は億単位) (保守費用も高額)	PCサーバ 1台 CPU 6コアx2 メモリ 48GB HDD SATA(2TB)x2 (初期費用は100万未満) (保守費用も安価)	PCサーバ 5台 CPU 6コアx2 メモリ 48GB HDD SATA(2TB)x2 (初期費用は500万未満) (保守費用も安価)

開発生産性

usp lab.

・COBOLの場合

7JOB、24処理ありますので、

3ヶ月程度かかると思われます。

・ユニケージの場合

コーディング　　：　5日

動作検証　　：　5日

パフォーマンス改善：　3日

・ユニケージエンジニア1名（経験5年）が

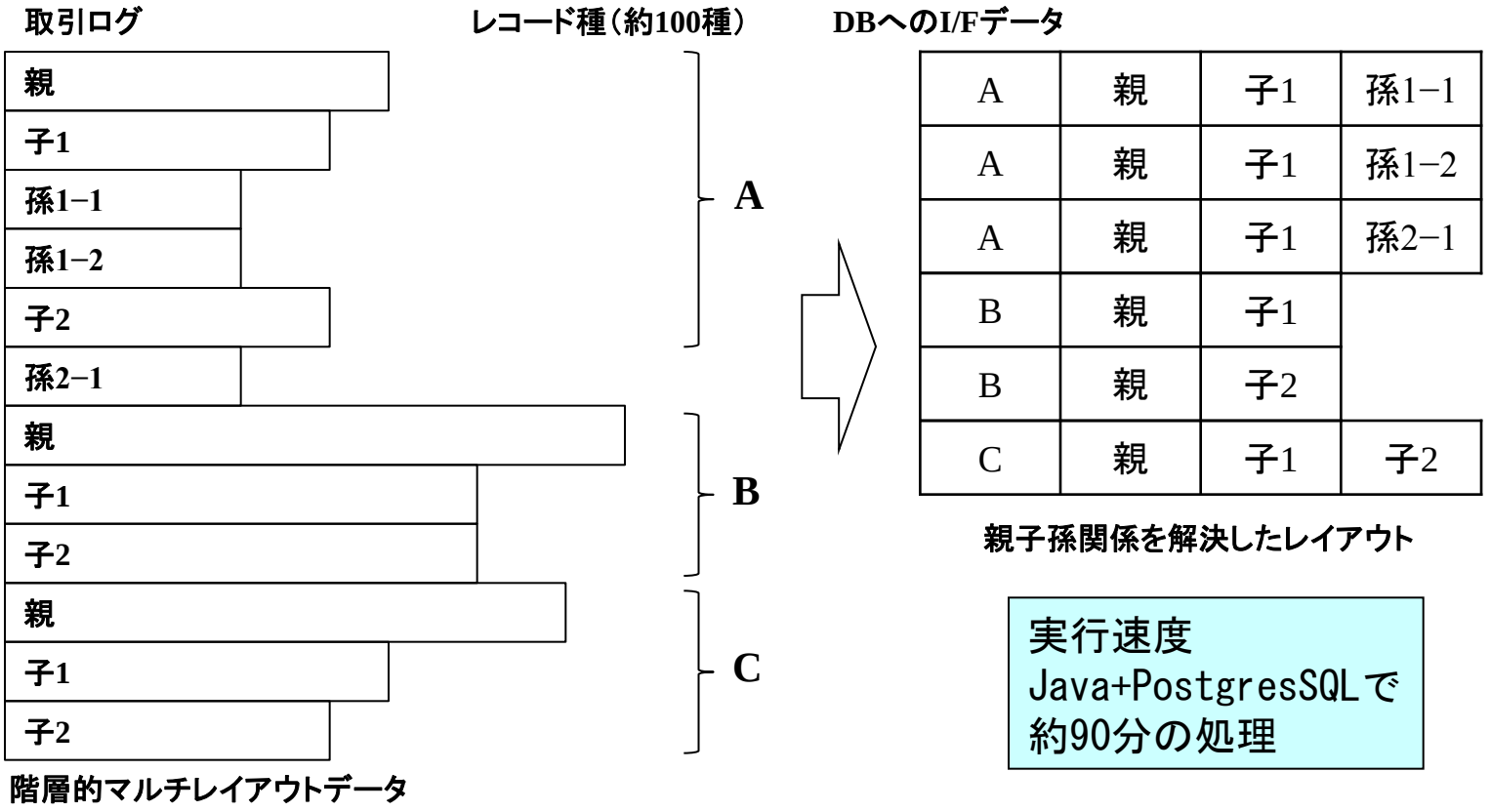
合計13日で開発しています。

	COBOL	ユニケージ
処理の数	7JOB、24処理	11シェルスクリプト
開発期間	3ヶ月（想定）	13日
ステップ数	3，645ステップ	981ステップ

②複雑なETL（某証券会社）概要

usp lab.

ユニケージ開発手法を使って
取引データ蓄積データベースにロードできるフォーマットの変換処理を行います。
処理時間について比較を行っています。



A	親	子1	孫1−1
A	親	子1	孫1−2
A	親	子1	孫2−1
B	親	子1	
B	親	子2	
C	親	子1	子2

親子孫関係を解決したレイアウト

実行速度
Java+PostgresSQLで
約90分の処理

Universal Shell Programming Laboratory

処理速度

usp lab.

開発・実験環境

コンピュータ	デスクトップPC （Intel Core i7プロセッサ, メモリ16GB）
オペレーティングシステム	FreeBSD 9.0 Release#0
シェルコマンド	usp TUKUBAI ビジネスバージョン

アプリケーション	内容	処理件数	実効ステップ数
PROCESS-MASTER	トップシェル		29
PROCESS-001	例外処理1	8,327 件	8
PROCESS-002	例外処理2	117,838 件	9
PROCESS-003	例外処理3	81 件	11
PROCESS-004	例外処理4	5,028 件	19
PROCESS-005	例外処理5	332 件	14
PROCESS-006	通常処理	27,614,260 件	6
		29,015,393件 （4.36Gバイト）	96

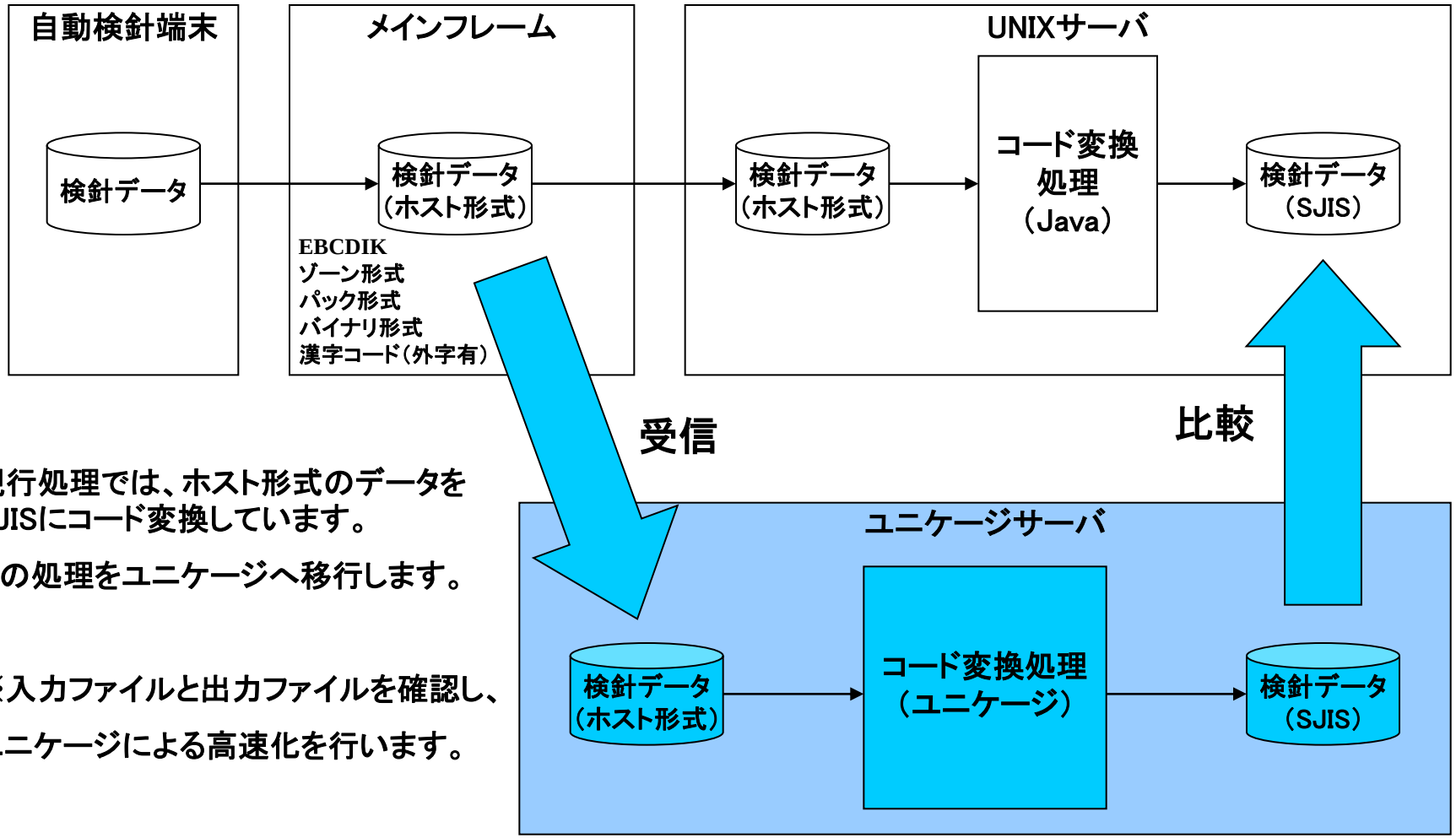
実行速度
real ： 91.58sec
User ： 132.85sec
sys ： 22.53sec

Universal Shell Programming Laboratory

③複雑なETL（某電力会社）概要

usp lab.

ホストデータのコード変換（ホスト形式→SJIS）を行います。



現行処理では、ホスト形式のデータを
SJISにコード変換しています。

この処理をユニケージへ移行します。

※入力ファイルと出力ファイルを確認し、
ユニケージによる高速化を行います。

Universal Shell Programming Laboratory

処理速度

usp lab.

2G、5G、10Gの3種類のデータ量で計測を行いました。

使用したサーバは以下の通りです。

・Java　　：　HP-UX, Itanium 1.60GHz 2core, 4GB

・ユニケージ　：　FreeBSD, Core i7 4core, 16GB, SATA(2TB)

データ量	2GB（7,240,555件）	5GB（18,095,303件）	10GB（36,178,437件）
Java	3時間7分53秒	7時間30分 （想定値）	15時間 （想定値）
ユニケージ	43．411秒	1分49．085秒	4分16．906秒
差異	11273秒/43.411秒 →259倍	27000秒/109.085秒 →247倍	54000秒/256.906秒 →210倍

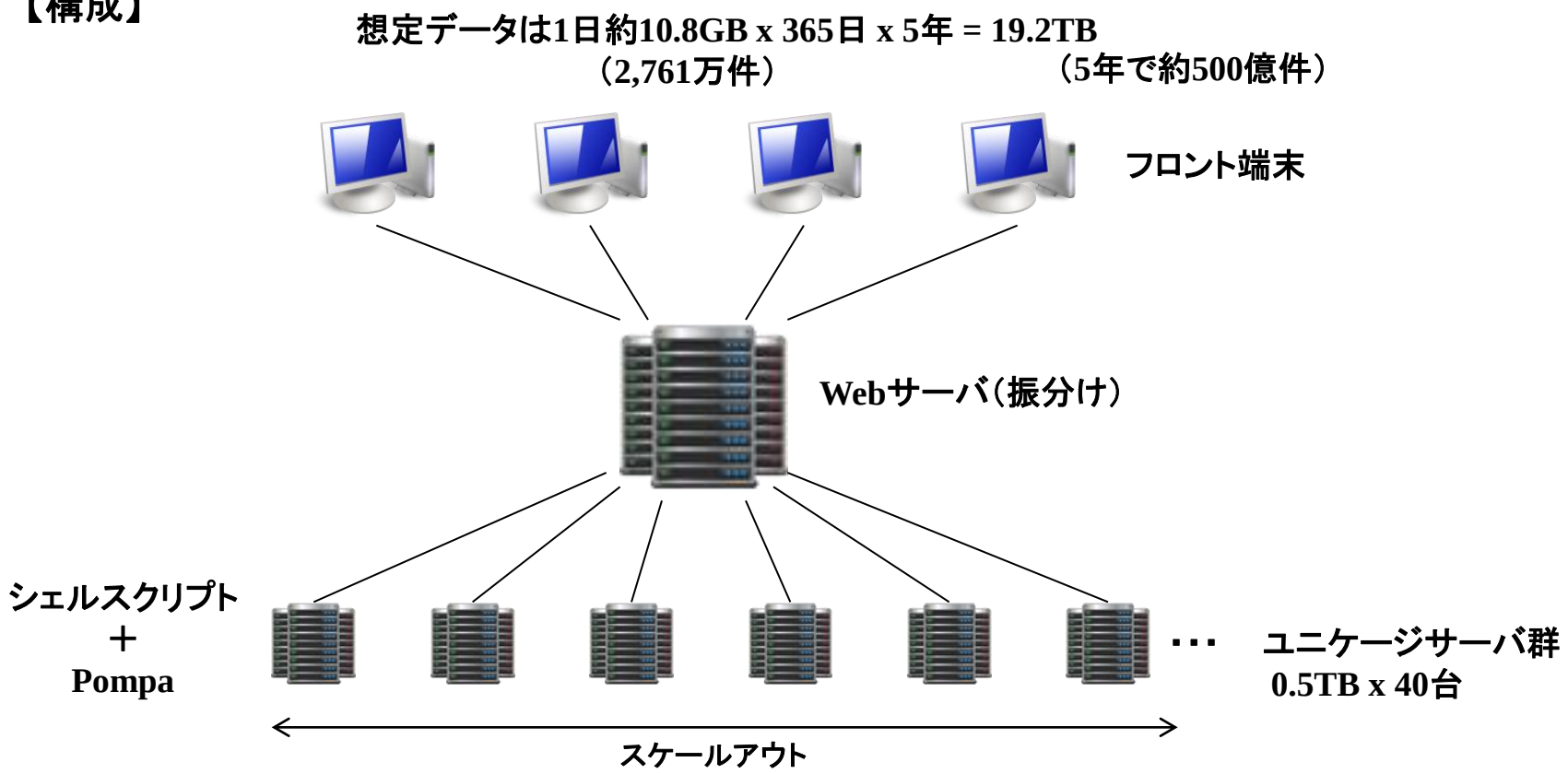
④大量データ検索（韓国大手検索サイト）概要

usp lab.

大規模サイトの検索ログ分析

検索文字列・ユーザアクセス元IPアドレス等を軸に分析

【構成】



Universal Shell Programming Laboratory

SQLとシェルプログラミング(1/2)

B3 ：C_QUERY_NOSP, C_USER 毎の行数を数える
B4 ：C_USER 毎の行数を数え、行数が30以上を出力
B5 ：C_DATE と C_USER を条件に C_QUERY_NOSP を一覧
B6 ：C_REQ_FRM 毎の行数を数え、行数の降順で出力
B7 ：C_CONNECTION 毎の行数を数える
B8 ：C_DATEとC_CONNECTIONを条件に1行以上のC_QUERY_NOSP毎行数を数える
B9 ：C_CONNECTION'X'のみの500以上C_QUERY_NOSP毎の行数を数える
B10：C_QUERY_NOSP毎の3以上のC_SESSION1 のユニーク数を数える
B11：別の指定日にない C_QUERY_NOSP毎の行数を数える
B12：3以上の C_IP 毎の行数と C_QUERY_NOSPのユニーク数を数える

SQLとシェルプログラミング(2/2)

usp lab.

2.シェルプログラミングの例
シェルプログラミングとSQLの対応例です。

```
B3【SQL】:
select C_QUERY_NOSP, C_USER, count(*)
 from SEARCHLOG
 where C_DATE='2006-09-18'
 group by C_QUERY_NOSP, C_USER;
```

```
B3【USP】:
cat ${lv3d}/L3.DAY     |
awk '$4=="20060918"    |
self 23 16             |
dsort key=1/2          |
count 1 2
```

```
B9【SQL】:
select A.q1, A.cnt1 as a1, B.cnt2 as a2 from
(select C_QUERY_NOSP as q1, count(*) as cnt1
  from searchlog
  where C_DATE='2006-09-18' and C_CONNECTION='X'
  group by C_QUERY_NOSP having count(*)>500 ) A,
(select C_QUERY_NOSP as q2, count(*) as cnt2
  from searchlog
  where C_DATE='2006-09-18' and C_CONNECTION<>'X'
  group by C_QUERY_NOSP) B
 where A.q1=B.q2 order by a1 desc, a2 asc;
```

```
B9【USP】:
cat ${lv3d}/L3.DAY                  |
awk '$4=="20060918"&&$14!="X"'      |
self 23                             |
dsort key=1                         |
count 1 1                           > $tmp-b
cat ${lv3d}/L3.DAY                  |
awk '$4=="2006-09-18"&&$14=="X"'    |
self 23                             |
dsort key=1                         |
count 1 1                           |
awk '$2>500'                        |
join1 key=1 $tmp-b -                |
sort -k2,2nr -k3,3n
```

処理速度

usp lab.

開発・実験環境

コンピュータ	デスクトップPC （Intel Core i7プロセッサ, メモリ16GB）
オペレーティングシステム	FreeBSD 9.0 Release#0
ストレージ	SATAハードディスク１台
シェルコマンド	usp TUKUBAI ビジネスバージョン

	対応SQL	実行時間(最小)	実行時間(最大)	実行時間(平均)
#01	B3	1.132	1.357	1.235
#02	B4	0.139	0.140	0.139
#03	B5	0.002	0.003	0.002
#04	B6	0.002	0.002	0.002
#05	B7	1.154	1.155	1.154
#06	B8	0.030	0.030	0.030
#07	B9	2.673	2.898	2.748
#08	B10	1.440	1.440	1.440
#09	B11	4.760	4.766	4.763
#10	B12	0.006	0.006	0.006

⑤まとめ

問題1 【パフォーマンスの低下】
　データ量の増加や度重なる業務ロジックの変更により、
　処理が徐々に遅くなり、業務に支障が出る。

問題2 【コスト】
　高速な専用ハードウェアや高機能なミドルウェアが必要となり、
　初期費用や維持費用が高い。

【問題の背景】
データの精度向上や
過去データの蓄積が進み、
データの件数が増えているが
RDBは大量データの扱いが
得意ではない。

【一般的な対応】
　最新の高速な専用ハードウェアや高機能なミドルウェアを導入する。
→パフォーマンスは改善されるが、その分コストが高い。
　ソフトウェアを最新技術（Hadoop等）で作り直す。
→高コストかつ、技術者の確保や育成が困難である。

なぜユニケージは高速か？ 1/2

usp lab.

①オーバーヘッドの大きいミドルウエアを組み込みません

DBやランタイムなどのミドルウェアを使わず、OSの基本機能のみを使います。この観点ではUNIX・Linux系ではFreeBSDがカーネルコードがコンパクトで、必要な周辺ソフトはPORTSコレクションから選定導入できなど、優れています。

②高度にチューニングされたusp Tukubai コマンド群

シェルスクリプトで使用するコマンドを基本的にはC言語で自作し、直接メモリやCPUをコントロールします。SIMD命令をインラインで記述するなど、相当チューニングされています。そのレベルは、通常のjavaプログラマがつくるコマンドの数十倍のパフォーマンスがでます。（コンパイル後のアセンブラのコード量に圧倒的な差が出ます）

③パイプラインによる並列処理

シェルスクリプトはUNIX系の独自技術であるパイプを簡単に利用できます。パイプラインで usp Tsukubai コマンドを接続することで並列処理が実現し、処理が高速化します。某証券会社の取引データ3000万件の処理は、16 core マシンのサーバーを使って全CPU平均使用率を95%にまで高め、従来スピードの60倍になりました。

Universal Shell Programming Laboratory

なぜユニケージは高速か？ 2/2

usp lab.

④ ush

シェルそのものもオーバーヘッドをなくすため、ush というash ベースの自作シェルを作成しています。同じシェルスクリプトでも標準的な bash の 1.7 倍のパフォーマンスを発揮します。ush は進歩を続けており、パイプの実装を、mmap （カーネルメモリ）に変更し、ID 渡しに変更するなど、さらなるスピードアップを目指しています。

⑤ pompa

大量データの高速検索を行うために、テキストファイルのディレクトリーツリー分割とメモリ上のキャッシュコントロールを行なっています。(pompa technology)パス名に検索キーを埋め込み、OSとuspTsukubaiコマンドの２段階検索で高速検索を実現します。この技術により、１０テラバイトのログデータ（韓国大手検索サイト）の複雑な検索を、高価なアプライアンスを使うことなく、０．１秒レベルで実現しています。