GenLM:License Management for Grid and Cloud Computing Environment

村田研究室
09M38370 元田　剛史

1

参考文献

- "GenLM:License Management for Grid and Cloud Computing Environment"
  - Mathias Dalheimer & Franz-Josef Pfreundt, Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid - Volume 00,Pages 132-139, 2009
- "Towards SLA Based Software License Management in Grid Computing"
  - Jiadao Li, Wolfgang Ziegar, Oliver Waldrich & Daniel Mallmann, CoreGRID Technial Report Number TR-0136, June25, 2008
- "Basic Spin Manual"
  - http://www.asahi-net.or.jp/~hs7m-kwgc/spin/Man/Manual_japanese.html
- "SEDA: An Architecture for Highly Concurrent Server Applications"
  - http://www.eecs.harvard.edu/~mdw/proj/seda/

2

目次

3

背景

Software license managementとは
- 各企業の開発したソフトウェアに対する利用者のアクセス管理を行うこと。
- Independent Software Vendors(ISVs)のビジネスの中核を担う重要なシステムである。

- 近年GridやCloudが科学の分野や商業的分野で注目を集めている。
- 様々なGrid用ミドルウェアが利用可能な中、Gridで利用可能なSoftware licenseを管理するインフラが整備されていない。

4

Gridにおけるlicensingへの要求

ユーザ
・Grid又はCloud上でソフトウェアを利用する人。
求めているもの
・ライセンス管理はGridのジョブ管理の一環として行われ、ユーザは意識しなくて済む。

リソース提供者
・Grid 又は Cloud環境の提供者。
求めているもの
・ユーザがアプリケーションについて多くの選択肢を持てるようなインフラ。

ISVs
・ソフトウェアの提供者。
求めているもの
・企業の経験や知識の結晶であるソフトウェアをライセンス認証なく使ってほしくない。

5

現在Gridで用いられているライセンス管理

- FlEXnet
  - pricing models
    - ある一定期間ごとにライセンスの更新が必要
    - 永久に利用できるがアップグレード時にライセンスの再取得が必要なもの
    - ソフトウェアを同時に利用する利用者の人数によって課金
    - ソフトウェアを利用するマシン又はサーバーごとに課金
    - CPU又はCPUコア数ごとに課金
    - ソフトウェアを使用する利用者ごとに課金

    ・・・等

  ソフトウェアごとにpricing modelは異なり、ソフトウェアのバージョンごとに異なる場合もある。

6

現行のライセンス管理における問題点及び要求

- CPU、PCごとの課金はGrid環境では非常に高額になる。
- 最大利用時のライセンスを常時保持しているのでは金がかかるので、最低限のライセンスを取得し必要な際に必要な分のライセンスを取得したい。

⇒オンデマンドなライセンスの獲得

7

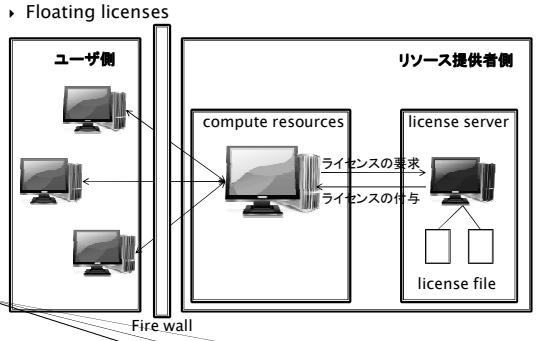
現在Gridで用いられているライセンス管理

- license model
  - Node locked licenses
    - 特定マシン・又は特定のマシン上のみでアプリケーションを利用可能。
  - Floating licenses
    - 保有する最大数までならば、ローカルネットワーク上のユーザはアプリケーションを利用可能
  - Demo licenses/evaluation licenses
    - 以下の特徴を持つライセンスを取得可能
      - 機能に制限がある
      - 利用回数に制限がある
      - ある期間で消滅する
  - Mobile licensing
    - ネットワークに常時接続していないノートパソコンの上でもアプリケーションを一定期間利用可能
    - 同じネットワーク上の他の端末でもライセンスを利用可能
    - ライセンスを取得し、後ほどそのライセンスを用いてネットワークにつながっていない環境でアプリケーションを利用可能

8

現在Gridで用いられているライセンス管理

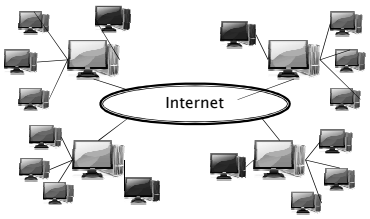
- Floating licenses



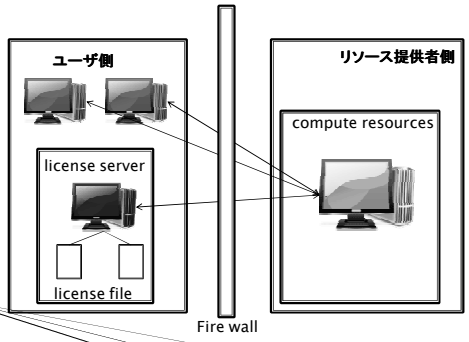
9

現行のライセンスモデルにおける問題

- ライセンスの利用が、Gridの管理上の範囲に限定される。
  - 利用するGridごとに異なるライセンスを取得する必要がある



FLEXnetの改善



11

現行のライセンスモデルにおける問題

- ジョブが開始されるまでにライセンスが発行されず、ジョブが失敗することがある。
- ライセンスサーバに対するアクセスについて
  - 認可におけるアクセス制御はIPアドレスを用いて行われている。その為、同じIPアドレスを割り当てられたユーザは利用出来る出来ないにかかわらずライセンスサーバへアクセスできる。
- 計算ノードとライセンスサーバはFirewallやNATで守られている為、全ての計算機上で正常に動作するシステムは難しい。

GenLMで必要なもの

機能要件

・存在する全てのライセンス規約をサポートすること。
・CPUごとのライセンスに加え、オンデマンドでライセンス取得を行えるようにする。
・ライセンスの"mobility"
計算の実行環境に依らずライセンスを利用できる。

非機能要件

既に存在しているソフトウェアパッケージにも簡単に導入可能

GridやCloudへのサポートが標準で組み込まれている

ライセンスは欲しい時に手に入る
・システムが原因でライセンスが手に入らないといったトラブルが無い。
システムがセキュアである
・ライセンス要求なしにライセンスが与えられることはない。
・ソフトウェアだけではライセンスチェックをごまかすことが出来ない。

13

GenLM

GenLMは**入力データにライセンスを与える**

- ユーザ・プロバイダ・ISVにそれぞれ1つずつコンポーネントを設置。

License token
GenLM license verifierがライセンスをチェックする際に必要な情報が入っているファイル。

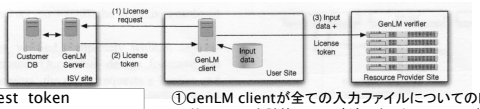
ISV
The GenLM Server
・要求が送られてきた際にライセンスに発行するサーバ

ユーザ
The GenLM client
・ユーザ又はソフトウェアがlicense tokenを手に入れるために用いる。

プロバイダ
The GenLM license verifier
・ISVの計算ソフトに含まれている。与えられたライセンスが、入力されたジョブに対し正当かどうかをチェックする。

14

ライセンスの取得について



Request token
・ライセンス規約
・全ての入力ファイルのハッシュ値

License token
・ライセンス規約
・全ての入力ファイルのハッシュ値
・署名

①GenLM clientが全ての入力ファイルについての暗号学的ハッシュを計算。ユーザが要求したライセンス規約をRequest tokenに格納する。
Request tokenとユーザのX.509証明書をGenLM Serverへ送る。

②GenLM Serverはトークンからライセンス規約とユーザ固有の情報を得る。
ライセンス規約はPolicy Pluginへ送られ、各ISVのライセンス規約に基づいて課金の有無等を判断する。そして、条件に基づいてRequest tokenにライセンスを与える。

③入力データと合わせて、License tokenがプロバイダへ転送され、ジョブを実行する際にLicense tokenを検査する。
ⅰ)署名を検査する。
ⅱ)入力データのハッシュ値を計算し、ライセンストークン内のハッシュ値と照らし合わせる。

15

暗号化について

Request Token
RT= (LT,H_I)

RT: Request Token
LT: ライセンス規約
I:入力ファイル
H_I:入力ファイルIのハッシュ値の集合

i∈Iについて
$H_i = h^F(i)$
h^F :暗号学的ハッシュ関数

GenLM ServerはRequest tokenを非対称キーで暗号化し、RTに加える。

暗号化
$sig(RT) = e_s(h^S(RT))$
h^S:署名用hash関数

非対称キー(p,s)
p: 公開鍵(public key)
s: 秘密鍵(secret key)

License Token
T= (RT, sig(RT))

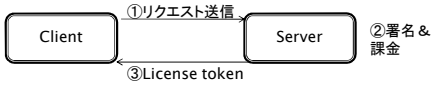
license verifierは公開鍵で復号、ローカルで計算したRTのハッシュ値と比較。

署名がGenLM Serverによるものかを確認
$h^S(RT) \stackrel{?}{=} e_p(sig(RT))$

16

プロトコルの設計

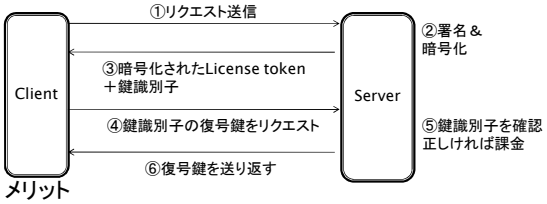
- Simple request-reply schema



- "Byzantine Generals" Problem
  - ③におけるLicense tokenの送信がネットワークの不具合が原因で失敗することがある。
    - 問題1:Clientはライセンスが無いのに金を払わなければならない。
    - 問題2:License tokenを正常に受け取っているのに、受け取っていないと主張し別のファイルをリクエスト送信できる。
      ⇒ライセンスを取得せずソフトウェアを利用できる。

17

プロトコルの設計



- メリット
  - ⑤で暗号化されたLicense tokenがClientに届いていることが確認できる。
  - ⑥で復号鍵が失われても、鍵識別子を確認すれば復号鍵を送り返せるので、新たにファイルにライセンス付与する必要が無い

18

設計したプロトコルの検証

- SPINを用いてモデルの検証を行った。

SPIN
並行システム（中でもデータ通信プロトコル）の論理的一貫性を検証するツール
・デッドロックの有無、意図しないメッセージを受け取らないかどうか、プロトコルが正しくデータを転送するという命題等を数学的に証明可能

- 結果、以下のことが証明された。
  - Deadlock, Livelockは含んでいない。
  - もし、Clientが復号鍵を手に入れたら、Serverはいずれ課金する。
  - もし、Clientがライセンスを取得したら、Serverはいずれ課金を行う。
  - もし、Serverが課金していなければ、Clientが復号鍵を手に入れることはない。
  - もし、サーバーが復号鍵を送信したら、Clientはいずれ課金される。

19

GenLM License server内の実装

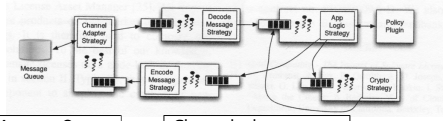
- SEDA(Staged Event-Driven Architecture)を使用
  - インターネットサービス用の開発プラットフォーム
  - マルチコアCPUに対応
- SEDA
  - SEDAは独立に動作するStageによって構成されている

Stage
input queue
・work itemを貯めておく
threads
・work itemをdequeueしてStageのstrategyに適用させる

strategy
work itemに対して適応させるアルゴリズムを定義したもの

20

GenLM License server内の実装



Message Queue
メッセージの送受信を行う

Channel adapter stage
受け取ったメッセージをDecoding stageに送る

Decoding stage
受け取ったメッセージ(ワイヤ・フォーマット)をサーバ内部で利用するフォーマットに変換。Application logic stageへ送る。

Application logic stage
暗号化する際はデータをCryptographic stageへ送る
ライセンスを許可するかどうかを判断するためにPolicy Pluginへキューを投げる
Clientにメッセージを投げる際はメッセージをEncoding stageへ送る

Policy Plugin
各企業の規約に基づいて、プロトコル上の異なるstageにおけるApplication logic stageで呼び出すべき関数を実装できる。

21

Performance Consideration

・ネットワークサーバの処理について

ネットワークサーバは多くの場合loadが増えれば応答時間も増える。

・本システムはSEDAを使っているため高負荷に強い。

・Message Queueがある為、サーバを増やして処理能力を高めることも可能。

・Message Queueにqueueを貯めておいて、負荷に耐えられるサーバがデータを取りに行くということもできる。

22

Performance Consideration

・サーバ上での暗号化処理速度に関するテスト

方法:

ランダムに100MB, 1GB, 10GBのファイルを作成

各ファイルに対する暗号化の処理速度を、異なるファイルシステム上で計測

"Hercules" cluster system
CPU:dual Intel Xeon 5148LV
メモリ:8GB
HD:80GB

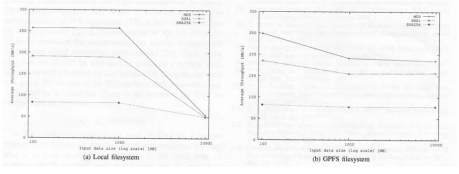
Local file system
ローカルHDを用いたファイルシステム
スループット 55MB/s

GPFS file system
分散共有ファイルシステム
スループット 450MB/s

23

Performance Consideration

- 結果



- Local filesystem
  - メモリにデータが乗る1GBまでは高スループットだが、10GBになるとスループットが大きく落ちる。
- GPFS filesystem
  - 10GBでも高いスループットを達成した。

24

まとめ

- GenLMはGrid及びCloud上で動くLicense management frameworkである。
- GenLMは入力データに対してオンデマンドにライセンスを与える。
- GenLMは13pで示した機能要件を満たしている
  - ライセンスを”mobile”可能
  - 現在存在しているライセンス規約はPolicy Plugin内でサポート可能
  - システムはsecureかつ暗号化方式は変更可能

25

感想

- ISVでライセンスサーバを用意する必要がある本モデルは果たしてISVに受け入れられるのか？
- オンデマンドにライセンスを取得する為に入力データにライセンスを与えるという考えは、今までになく非常に面白いと感じた。

26