2017.2.7 【DMM GAMES主催！】「複雑・大規模webサービスを支える技術勉強会」

大規模になったサービスでやるべき基本的なこと

XFLAG STUDIO

ゲーム開発部 SREグループ　清水 勲 @isaoshimizu

About me

- 清水 勲 @isaoshimizu
- 2011.8-2014.3 SNS mixiの運用
- XFLAG スタジオ
  - 2014.4-2016.6 サーバーエンジニア
  - 2016.7- SRE 主に国内向けモンスターストライクを支えるお仕事
  - 他にもモンストスタジアム、ブラナイDASHなど
- 好きなもの
  - Linux、MySQL、nginx、Memcachedなどのミドルウェア、Go、クラフトビール 🍺

直近の発表資料

- 2016.3.1 モンストを支えるインフラの今とこれから

  https://speakerdeck.com/isaoshimizu/monsutowozhi-eruinhurafalsejin-tokorekara

- 2017.1.30 SREグループができてこの半年間やってきたこと

  https://speakerdeck.com/isaoshimizu/sregurupugadekitekofalseban-nian-jian-yatutekitakoto

サービスの規模が大きくなるということ

サービスの規模が大きくなるということ

- アクセスが増える
- 負荷との戦いが始まる
- サーバが増える
- 関わる人も増える

負荷を可視化する

監視・モニタリング

- 「なんとなく負荷高そう」という状態はとても危険
- 便利なツールが増えてきているのでちゃんと活用する
    - Nagios, CloudForecast, Cacti, Ganglia, Munin, Sensu, Zabbix, Kurado, Prometheus
    - Mackerel, Datadog, CloudWatch, NewRelic, PagerDuty, Pingdom
    - Fluentd+Elasticsearch+Kibanaでログからグラフを生成するのも効果的
- グラフは毎日確認する。理解できないグラフは生成・表示しても意味がない。
    - 毎月負荷状況をサマライズしておくと振り返りや今後の対策方針が作りやすい
- 見えないもの、測れないものはリスク
- サービス連動の企画やイベントは早めに共有して備える

スケールアップかスケールアウトすべきか

スケールアップ or スケールアウト

- スケールアップ
  - 性能を上げて負荷対策
  - スケールできる上限が決まってしまう。さらなる負荷増への対応は厳しい。
  - PCI-ExpressなioMemoryやNVMe SSDを使ってもダメな時はダメ。
- スケールアウト
  - LB配下のアプリケーションサーバは増やすことでCPUリソースを負荷分散
    - DBへの接続とアクセスが増えるので要注意
  - DBにおいては、スケールアウトできると、中長期的な負荷対策として有効
  - 一時的な負荷増への対策であればスケールアップも効果的

トラフィックの増加

トラフィックの増加

- クラウドの場合
  - インスタンスタイプによって上限が異なる（ベンチマークとっておくと安心）
- 自社DCを使ってる場合
  - NICとスイッチの帯域上限（サーバーがどのラックにあるかを意識）
  - 回線の冗長性（方系が落ちても許容できる帯域設計）
  - NICの冗長性
    - Bonding: コネクション単位で分散されるのでNICx2で2倍の帯域とは限らない
- TCPの再送回数の視覚化
  - 特にクラウドではレイテンシや帯域の変化があるので要注意

サーバー/インスタンス数の増加

サーバー/インスタンス数の増加

- デプロイ、プロビジョニングの効率化、自動化
  - Capistrano, AWS CodeDeploy, Stretcher + Consul
  - Chef, Ansible, Puppet, Itamae
- ベースとなるイメージ（AMIなど）構築の自動化
  - Packer便利
- 台数に依存しないオペレーションが理想系

データベース クエリ数の増加

データベース クエリ数の増加

- ボトルネックはREADかWRITEか
- データベース、テーブルの単位でサーバを分割
- シャーディング（IDやハッシュ値の剰余で分割）
- キャッシュ（Memcached, Redisなど）の活用
- ストレージのIO限界の見極め（カタログスペックを信用せず計測 する）
  - tmp table, file sortなどが発生していないか
  - Dirty Pageやwaitの発生頻度をグラフから知る
- バッファプール利用率の状況

データベース サイズの増加

データベース サイズの増加

- 一度溢れてしまうと対処が困難
- テーブルのデータ、インデックスサイズ
- バッファープールのキャパシティ
  - メモリを増やしてプールサイズを広げるという手
- バイナリログの増加。expire_logs_daysの調整（MySQLの場合）。
- AUTO INCREMENTの上限値に注意（intは約21億、bigintは約922京）
  - intだとテーブルによってはあっという間に達してしまうので注意
  - 閾値を設定して監視しておくと安心

人員の増加

人員の増加

- アカウント
  - クラウドのアカウント、Linuxユーザーアカウント、GitHubのアカウント
  - AWSの場合、CloudTrailを使った監査ログ
  - LDAPなどでログインユーザー、ホストの制限（セキュリティの担保）
  - プロジェクトへ新たにジョイン＆抜けるメンバーへの対応（チェックリストのテンプレート）
- アラート
  - 当番制、エスカレーション方法などの確立、PagerDutyの活用
- ノウハウや手順の共有
  - Wikiの活用（例: GitHub Wiki, Qiita Team, Crowi）
  - 構築や障害発生時の手順、ルールをドキュメント化（インシデント発生時に冷静に対処できるように）
  - 4 eyesでの作業確認して事故防止（例: DNS更新、Terminate、Service Stopなど）
  - ChatOpsで作業できることを増やす

まとめ

まとめ

- 規模にかかわらず基本をしっかりと
- 起きていることを正しく把握すること、イメージできること
- 規模に比例して作業量が増えないようにすること
- 人数が増えたときに作業負荷が分散されるようにノウハウや手順を展開すること
- 大規模じゃなくてもやったほうがいいことは多い

- 仲間募集中 https://xflag.com/recruit/

Thank you!