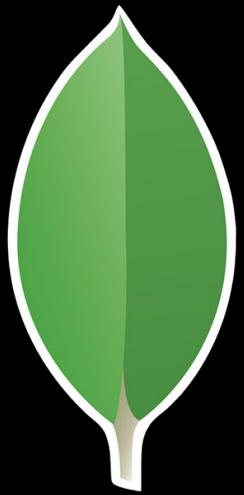
SRG Study #1

MongoDBについて

MongoDBの運用時における障害事例とその対応、対策について

株式会社サイバーエージェント
技術本部 サービスリライアビリティグループ
小林 健太郎



アジェンダ

- はじめに
- MongoDBの障害事例と対応、対策
  - VM/ディスク障害・予定されたメンテナンス等
  - メモリ不足による影響
  - シャードクラスターとバランサー
  - インデックスまわり
  - レプリカセットの切り離し失敗
  - mongosまわり
- MongoDB4.4での機能強化

- **はじめに**
- MongoDBの障害事例と対応、対策
  - VM/ディスク障害・予定されたメンテナンス等
  - メモリ不足による影響
  - シャードクラスターとバランサー
  - インデックスまわり
  - レプリカセットの切り離し失敗
  - mongosまわり
- MongoDB4.4の機能強化

MongoDBとは

- MongoDBは、開発とスケーリングを容易にするために設計された Document Database

ざっくり言うなら、RDBMSからjoin相当の機能とトランザクションを削ったもの

（4.2からはシャード間トランザクションも実装された）

RDBでも重くなるような複雑な機能を削り、代わりにノードの水平分散やデータの処理速度を早めることを可能にした

MongoDBとは

- アプリケーションを作りながらのスキーマ設計やサイジングが可能な為、スピード感のある開発に有用
- RDBMSを使っていた人でも使いやすいような構造
  - **RDBMSを補完することのできるDB**
- シンプルに高可用性を実現しているレプリカセット
- 水平スケーリングを実現する自動シャーディングとバランシング
  - **冗長性を持った負荷分散が可能**

MongoDBとは

- 反面、運用していく上で必要な知識も
  - どういった知識が必要なのかがわかりにくい

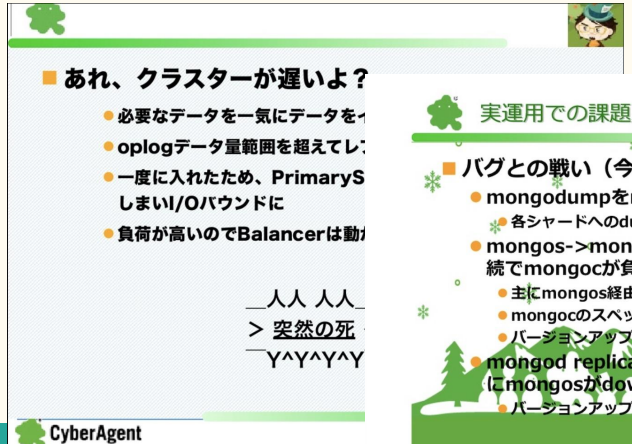
過去に発生したトラブルや障害事例から、機能に対する知識と対応、そして対策を見ていく

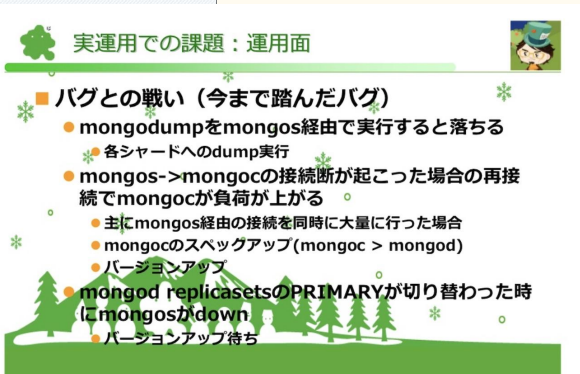
- はじめに
- **MongoDBの障害事例と対応、対策**
  - VM/ディスク障害・予定されたメンテナンス等
  - メモリ不足による影響
  - シャードクラスターとバランサー
  - インデックスまわり
  - レプリカセットの切り離し失敗
  - mongosまわり
- MongoDB4.4の機能強化

MongoDBの障害事例と対応、対策

- 過去のバージョンではよく発生したが、最近のリリースでは発生していないような障害は除く
- 主に、3.4系以降での障害と対応、その対策といった感じ

~~1.X系、2.X系初期の数々の地雷については、ググれば某くわのさんのスライドが出てくると思います~~



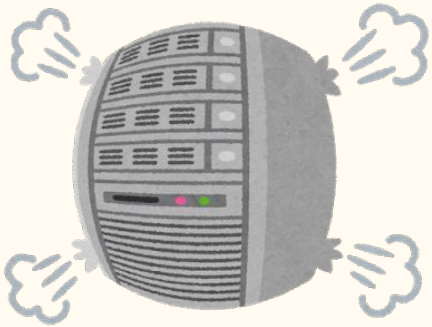


- はじめに
- MongoDBの障害事例と対応、対策
  - **VM/ディスク障害・予定されたメンテナンス等**
  - メモリ不足による影響
  - シャードクラスターとバランサー
  - インデックスまわり
  - レプリカセットの切り離し失敗
  - mongosまわり
- MongoDB4.4の機能強化

VM/ディスク障害・予定されたメンテナンス等

頻度：よくある

- すでにダウンしている場合
- ダウンが予定されている場合



Cycloud Info

障害情報 終了 tky02a openstack hostserver

ホスト障害のお知らせ

障害発生時間:2021/01/02 19:18:14
障害収束時間:2021/01/02 20:01:04

利用者各位

本日、以下の通り
ご利用中の方々には大変ご迷惑を

■障害内容
TKY02 ホスト障害

■影響範囲

EC2 instance reboot flexible maintenance scheduled

Back to list view

Details | Affected resources

Event data

Event EC2 instance reboot flexible maintenance scheduled	Start time December 18, 2020 at 12:15:28 PM UTC+9
Status Completed	End time December 18, 2020 at 12:15:28 PM UTC+9
Region / Availability Zone ap-northeast-1	Category Scheduled change
Affected resources 1	

VM/ディスク障害・予定されたメンテナンス等

- すでにダウンしている場合（1）
  - ダウンしたホストを再作成
    - この時、レプリカセット生成時と同じホスト名で再作成する
  - プロビジョニング後、mongodプロセスを起動すると自動で同期が開始
  - 同期が完了すると、自動でレプリカセットメンバに復帰

VM/ディスク障害・予定されたメンテナンス等

- すでにダウンしている場合（2）
  - この際、mongodの自動起動は行わないほうが（経験上は）良い
    - 同期開始と共に、レプリカセットメンバへの負荷が上がってしまう
      - 手動で任意のタイミングのほうが安心
    - Read Preferenceにて、secondaryを指定している場合、進行中のレプリケーションが完了するまで読み込みがブロックされる
      - MongoDB 4.0以降で改善

VM/ディスク障害・予定されたメンテナンス等

- ダウンが予定されている場合（1）
  - 対象がprimaryサーバである場合
    - secondaryに降格を実施
    - mongodプロセスを停止
    - ホストを再作成
    - プロビジョニング後、プロセスを起動
    - （自動で）再同期
    - （自動で）再投入

VM/ディスク障害・予定されたメンテナンス等

- ダウンが予定されている場合（2）
  - 対象がsecondaryサーバである場合
    - mongodプロセスを停止
    - ホストを再作成
    - プロビジョニング後、プロセスを起動
    - （自動で）再同期
    - （自動で）再投入

- はじめに
- MongoDBの障害事例と対応、対策
  - VM/ディスク障害・予定されたメンテナンス等
  - **メモリ不足による影響**
  - シャードクラスターとバランサー
  - インデックスまわり
  - レプリカセットの切り離し失敗
  - mongosまわり
- MongoDB4.4の機能強化

メモリ不足による影響

頻度：まれ

- オンメモリのreadアクセスについて（1）
  - wiredtigerエンジンになった現状でも、MongoDBではキャッシュ上にによく使われるデータ（ワーキングデータ）を保持し、高速なアクセスを実現している
  - インデックスとワーキングデータが RAMに収まる場合、MongoDBはメモリからすべてのクエリを処理する
    - つまり、非常に高速！
  - 逆に言えば、RAMに収まらない状況では低速に

メモリ不足による影響

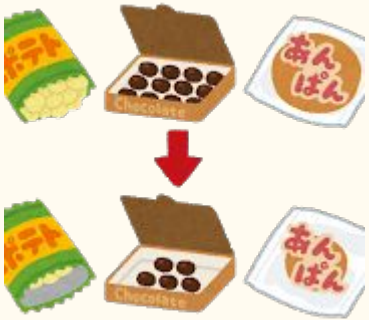
- オンメモリのreadアクセスについて（2）
  - キャッシュに追加のデータを読み込むのに十分なスペースがない場合、 WiredTigerエンジンはキャッシュ上からページを削除し、スペースを解放する
  - キャッシュ上にワーキングデータが乗り切らないような状況下において、全データアクセスするような処理ではpage in/outが多発し性能が出なくなる場合がある
- 対策
  - 監視の項目として、メモリの page in/outあたりを見ておくと、早期発見につながる

メモリ不足による影響

- メモリの設定について（1）
  - デフォルトのWiredTigerでのキャッシュサイズ値は、
    - 50% of (RAM - 1 GB), or 256 MB
  - 上記で計算した値より、各 shardが持つ実データが小さいことが望ましい
    - サービスの成長に合わせ、スケールアップ、もしくはスケールアウトが必要
  - もちろん、実際には全データが頻繁に使われることはないであろうから、あくまで指標の一つとして考慮

メモリ不足による影響

- メモリの設定について（2）
  - またデフォルト値は、マシンごとに 1つのmongodインスタンスがあることを前提としている為、1台のマシンに複数の MongoDBのプロセスが立ち上がる場合は、値を計算し減らす必要がある
  - このデフォルト値を設定するコンフィグ、storage.wiredTiger.engineConfig.cacheSizeGBはデフォルト値より値を大きくすることは推奨されていない
    - あくまで設定する場合は、小さい値を設定するときのみ



- はじめに
- MongoDBの障害事例と対応、対策
  - VM/ディスク障害・予定されたメンテナンス等
  - メモリ不足による影響
  - **シャードクラスターとバランサー**
  - インデックスまわり
  - レプリカセットの切り離し失敗
  - mongosまわり
- MongoDB4.4の機能強化

シャードクラスターとバランサー

頻度：よくある

- シャードとバランサーの動きに関連した障害は、MongoDBのメインの機能であるため、よくある
- 高機能である反面、ある程度挙動を理解していないと不調が起きてから設定の不備が発見されるパターンがあった

シャードクラスターとバランサー

- シャードキー未設定のまま機能リリース（1）
  - シャードキーを未設定、この場合はシャード化されていない状態、つまりプライマリシャードのレプリカセットにのみ該当のコレクションが存在する状態に
  - この状態のまま、該当コレクションを使用するイベントなどをリリースすると、リリースのタイミングからプライマリシャードの負荷がグングン上がっていき。。

シャードクラスターとバランサー

- シャードキー未設定のまま機能リリース（2）
  - 負荷がまだ耐えられそうであれば、オンラインにてシャード化を実行
  - 負荷が高すぎる場合は、メンテナンスに入れ該当コレクションをシャード化、手動にてチャンクの分割を実行後メンテナンスオープン
- 対策
  - 定期的なバッチにて、シャード化されていないコレクションがないかをチェック
  - アプリケーション側で、コレクションへの初回アクセス時にシャードキーの設定を必ず叩いたりも。。

シャードクラスターとバランサー

- 特定のシャードのみ高トラフィックが発生する（1）
  - チャンクを作成するのに十分なデータがない
  - シャードkeyの選定ミス
  - jumboフラグの存在

- レアケースも存在するが、特定のシャードでの負荷高騰は、ほとんどが上記のパターン

シャードクラスターとバランサー

- 特定のシャードのみ高トラフィックが発生する（2）
  - チャンクを作成するのに十分なデータがない
    - シャーディングは、各シャードのチャンク数がほぼ同じになるまで、シャード間でチャンクが移行することで機能
    - デフォルトのチャンクサイズは 64MB
    - シャードクラスター内のチャンクの不均衡が、移行のしきい値を超えるまで移行されない
    - デフォルトでは、3つを超える64MBのチャンクを作成するのに十分なデータがない場合、すべてのデータは1つのシャード（プライマリシャード）に残る

シャードクラスターとバランサー

- 特定のシャードのみ高トラフィックが発生する（3）
  - チャンクを作成するのに十分なデータがない
- 対策
  - 最初から高負荷が見込まれるが、データ量としては小さいマスターデータなどの場合、事前に手動でのchunk分割（Pre-Split）を実施する

シャードクラスターとバランサー

- 特定のシャードのみ高トラフィックが発生する（4）
  - シャードキーの選定ミス
    - シャードキーのカーディナリティが低い場合、MongoDBは十分な分割を作成できない
    - 極端な例だと、キーにmale/femaleなど2種類しかないものを用いたり
    - 適切なものを用いていない場合、各シャードがもつデータに偏りが生じ、特定のシャードのみ高トラフィックが発生する
      - 例えば、1月のアクセス2月のアクセス等、その月になると高騰するであろう値をシャードキーに選んでいる場合、12個に分散させても該当月を持つシャードが負荷高騰する。。
    - このパターンを修正するには、データを再シャーディングし、別のシャードキーを選択

シャードクラスターとバランサー

- 特定のシャードのみ高トラフィックが発生する（5）
  - jumboフラグの存在
    - チャンクの分散が特定のしきい値に達すると、バランサーはシャード全体にデータの分散を開始する
    - しかし、チャンク内のドキュメントの数が特定の数を超えると、 MongoDBはチャンクを移動できなくなる
    - 書き込みの件数が多いコレクションで、起きやすい
    - こういった場合、MongoDBはチャンクにjumboのラベルを付け、移動を保留してしまう

シャードクラスターとバランサー

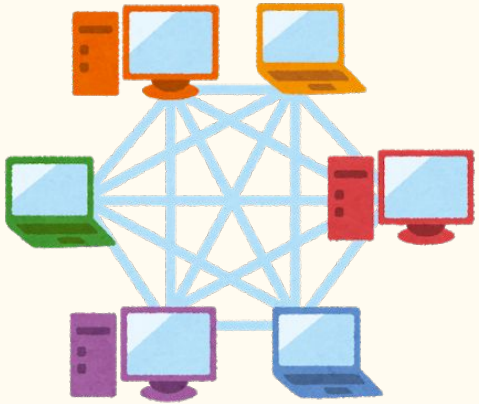
- 特定のシャードのみ高トラフィックが発生する（6）
  - jumboフラグの存在
    - チャンクサイズが指定されたしきい値を超えなくなった場合、 jumboフラグを自動的にクリアする
      - ただ、ひたすら書き込みが続くようなコレクションでは自動で消えることはほとんどない
    - 大抵の場合は、手動でフラグを消去する必要がでる
      - 定期的な確認と対応はどうしても必要になってしまう
      - 手動削除後、mongosに対してflushRouterConfigコマンドを発行すること（後述）

シャードクラスターとバランサー

- 特定のシャードのみ高トラフィックが発生する（7）
  - 最初はバランスが取れていたが、後で各シャードがもつデータに偏りが生じるパターン
    - クラスターから大量のデータを削除、または追加した場合
      - シャードキーに関して異なる分布になっている可能性
    - シャードキーのカーディナリティが低く、チャンクがこれ以上分割できなくなっている
    - データセットが、バランサーがデータを分散できるよりも速く成長している
      - データの増加率を考えると、バランシングウィンドウが短すぎる
        - バランシングウィンドウとは、バランサーの動作時間
          - 通常は常に動かすが、クラスタ全体のパフォーマンスに影響があるため（後述）、ピーク帯を避けるなどをしている場合がある

シャードクラスターとバランサー

- 特定のシャードのみ高トラフィックが発生する（8）
  - 最初はバランスが取れていたが、後で各シャードがもつデータに偏りが生じるパターン
    - シャード間のネットワーク接続が不十分
      - チャンクの移行が完了するまでに時間がかかりすぎる
        - ネットワーク構成とシャード間の相互接続を調査



シャードクラスターとバランサー

- 特定のシャードのみ高トラフィックが発生する（9）
  - シャードキー変更の手順（ MongoDB4.2以前）
    - MongoDBからすべてのデータを dumpする
    - 対象のコレクションを dropする
    - より理想的なシャードキーを用いて、シャードを構築する
    - 必要に応じて、chunkを事前に手動分割する
    - dumpしていたデータから対象コレクションに restoreする

シャードクラスターとバランサー

- 特定のシャードのみ高トラフィックが発生する（10）
  - シャードキー変更の手順（ MongoDB4.2以降）
    - 4.2からは、_id以外をシャードキーにしている場合、シャードキーのアップデートをオンラインで行うことが出来るようになった
    - 、、が、、試したことはないので割愛
    - 条件がいくつかあります

シャードクラスターとバランサー

- チャンクの移行がシャードクラスターのパフォーマンスに影響する（1）
  - バランサーによるチャンク移行により、その間のパフォーマンスが落ちてしまう場合の対策
    - バランシングウィンドウを制限して、ピーク時のバランシングを防ぐ
      - データのバランスが再び崩れないように、十分な時間が残っている必要がある

シャードクラスターとバランサー

- チャンクの移行がシャードクラスターのパフォーマンスに影響する
  - バランサーが常にチャンクを移行して、クラスター全体のパフォーマンスが常に悪い場合
    - チャンクサイズを小さくして、移行のサイズを制限する
    - クラスターの容量が不足している可能性があるため、 1つまたは2つのシャードを追加して、負荷を分散する
    - シャードキーによって、アプリケーションがすべての書き込みを 1つのシャードに行っているパターンも
      - この場合、データを書き込んだ直後にほとんどのデータをバランサーが移行しなければならないため、効率が悪い
        - 書き込みスケーリングを向上させるシャードキーを使用してクラスターを再デプロイ

- はじめに
- MongoDBの障害事例と対応、対策
  - VM/ディスク障害・予定されたメンテナンス等
  - メモリ不足による影響
  - シャードクラスターとバランサー
  - **インデックスまわり**
  - レプリカセットの切り離し失敗
  - mongosまわり
- MongoDB4.4の機能強化

インデックスまわり

頻度：たまに

- インデックスの貼り忘れ、もしくは効果的でないインデックスの利用（1）
  - 大抵は、機能リリース後の負荷高騰で判明する
  - インデックスについてはデータベースでは一般的であるため、最近は発生頻度が少ない
  - たまに起こるのは、テストに漏れてしまったりなどのレアなミス
- 生成時の挙動について説明していく

インデックスまわり

- インデックスの貼り忘れ、もしくは効果的でないインデックスの利用（2）
  - インデックスの生成（MongoDB4.2以前）
    - デフォルトではforegroundの動作になるため、対象のコレクションに lockをかける
      - この場合は大抵メンテナンスが必須に
      - インデックスビルドは高速、より効率的なインデックスデータ構造を生成できる
    - オプションで、background動作での生成も可能
      - オンラインで実施ができる
      - 遅く効率の悪い生成だが、read/writeアクセスが許可された状態

インデックスまわり

- インデックスの貼り忘れ、もしくは効果的でないインデックスの利用（3）
  - インデックスの生成（MongoDB4.2以降）
    - 生成プロセスの開始時と終了時に、対象コレクションに対してのみ排他ロックを取得
    - 残りの部分では、backgroundでの生成動作と同様
    - そのため、対象コレクション read/writeアクセスが許可された状態
    - ゆるいロック動作にもかかわらず、今までより効率的なインデックスデータ構造を生成
    - 4.2以降でのインデックスビルドは、最小でも以前の background生成と同等になり、更新がほとんどないコレクションに対してのインデックスビルドでは、 foreground生成と同じくらい高速になった

- はじめに
- MongoDBの障害事例と対応、対策
  - VM/ディスク障害・予定されたメンテナンス等
  - メモリ不足による影響
  - シャードクラスターとバランサー
  - インデックスまわり
  - **レプリカセットの切り離し失敗**
  - mongosまわり
- MongoDB4.4の機能強化

レプリカセットの切り離し失敗

頻度：まれによくある？

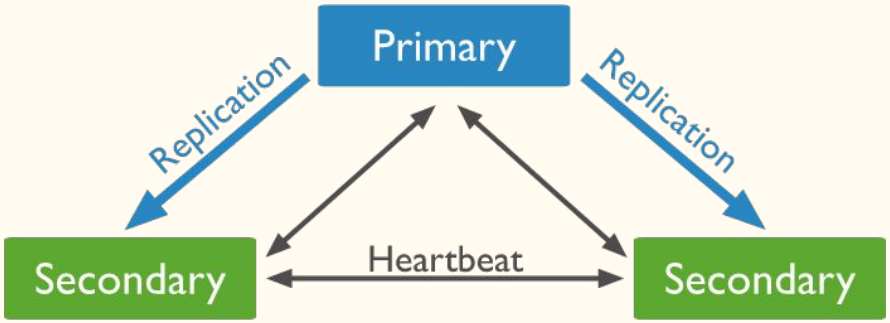
- レプリカセットメンバーの切り離し動作がうまく行かない（1）
  - 2年に1回くらい遭遇
  - 過去物理サーバでも2回ほど、MongoDBのバージョンが上がった最近の論理サーバでも 2回ほど遭遇している
  - エラーが突然高騰した際にはここも疑ってみる

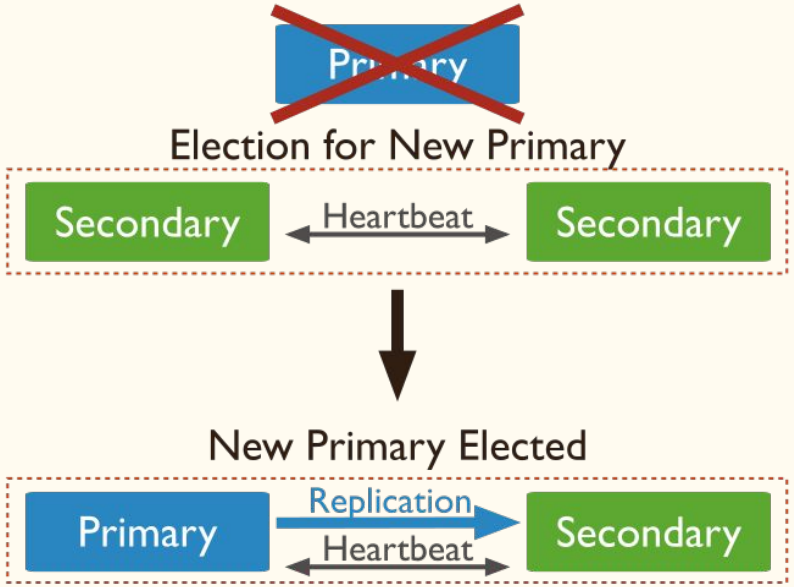
レプリカセットの切り離し失敗

- レプリカセットメンバーの切り離し動作がうまく行かない（2）
  - ホストの障害にて、primaryがダウン
  - primaryはsecondaryに降格のパターンも
  - レプリカセット内でなぜか正常に通知されず、primary不在に
  - read/writeアクセス不可で、このレプリカセットへのアクセスがエラーに

レプリカセットの切り離し失敗

- レプリカセットメンバーの切り離し動作がうまく行かない（3）
  - 本来の挙動





レプリカセットの切り離し失敗

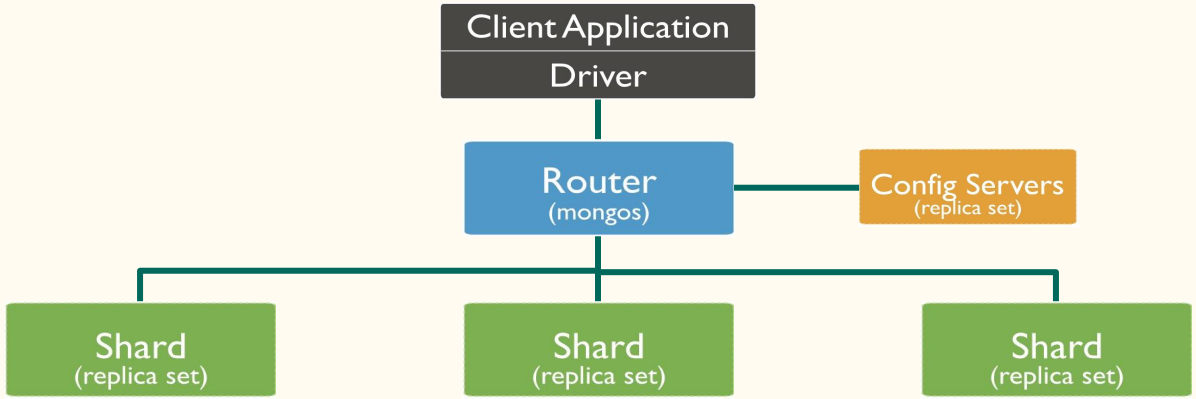
- レプリカセットメンバーの切り離し動作がうまく行かない（4）
  - 対応
    - 手動にてホストもしくはプロセスをダウン
    - 残るメンバ内にて昇格が行われ正常に
    - 原因ははっきりとは不明だが、中途半端なプロセスのダウンなどが原因で heartbeat(ping)だけを返していたり、物理サーバでは nicの故障で起きたりしていた

- はじめに
- MongoDBの障害事例と対応、対策
  - VM/ディスク障害・予定されたメンテナンス等
  - メモリ不足による影響
  - シャードクラスターとバランサー
  - インデックスまわり
  - レプリカセットの切り離し失敗
  - **mongosまわり**
- MongoDB4.4の機能強化

mongosまわり

頻度：たまに

- mongosが持つキャッシュが最新化されないための不具合（1）
  - mongos（ルーター）は、クエリと書き込み操作をシャードにルーティングする



mongosまわり

頻度：たまに

- mongosが持つキャッシュが最新化されないための不具合（2）
  - レプリカセット内にてprimaryの降格が発生した際、
    - read prefenceにて行っていたsecondaryからの読み込みが切り替わらずに、readでエラーが多発
    - 同じく書き込み先が何故か切り替わらずに、writeでエラーが多発
  - 上記のような事象がたまーに起きてる

mongosまわり

頻度：たまに

- mongosが持つキャッシュが最新化されないための不具合（3）
  - 対応
    - 今まで説明した、他の事象と複合的に起こっていることがほとんどだが、
      - アプリケーションドライバ
      - MongoDBのバージョン
  - ここらへんを疑って調査を行う

mongosまわり

頻度：たまに

- mongosが持つキャッシュが最新化されないための不具合（4）
  - 暫定対応
    - 即時対応が必要な場合、（他の不具合が見られない時）
      - mongosに対し、flushRouterConfigコマンドを発行
        - キャッシュされたルーティングテーブルを失効としてマークし、ルーティングテーブルにキャッシュの更新を要求させる
        - それでもだめなら、伝家の宝刀、再起動。。

mongosまわり

頻度：たまに

- mongosが持つキャッシュが最新化されないための不具合（5）
  - flushRouterConfigコマンドが必要な時
    - MongoDB4.2以前の場合、movePrimaryまたはdropDatabaseコマンドを実行後
    - MongoDB4.2.2以前（または4.0.14以前）の場合、分割できなかったチャンクからジャンボフラグを手動でクリアした後
      - ~~最近まで知らなかった、、！~~
    - db.collection.getShardDistributionコマンドを実行する前
      - シャードの情報を問い合わせるコマンド、これも知らなかった

- はじめに
- MongoDBの障害事例と対応、対策
  - VM/ディスク障害・予定されたメンテナンス等
  - メモリ不足による影響
  - シャードクラスターとバランサー
  - インデックスまわり
  - レプリカセットの切り離し失敗
  - mongosまわり
- **MongoDB4.4の機能強化**

MongoDB4.4の機能強化

- 簡単にトピックを書いておきます

MongoDB4.4の機能強化

- Query Language and Drivers
  - 複数のコレクションからのパイプライン結果を、ひとつの result setに結合する機能、$unionWith aggregation。
    - より深い調査と分析が実現できるようになる
  - aggregation pipelineとして実行される、独自の関数で MongoDBを拡張するカスタム集約式 $accumulator演算子と$function演算子
    - 独自のJavaScriptを実行し、MongoDBクエリでサポートされていない動作を実装できる
  - クラスター全体の読み取り分離と書き込み耐久性のための、 Global Read and Write Concerns
  - 新規にRustとSwiftのドライバー

MongoDB4.4の機能強化

- Scale-Out Flexibility and Performance
  - システムのダウンタイムなしで、いつでもシャードキーを定義および変更できる、 Refinable Shard Keys
  - シャード間で負荷をより均等に分散できるようになる、 Compound Hashed Shard Keys
  - 読み取り要求を複数のレプリカに送信し、最速のノードが応答し、すぐに結果を返すことにより、 p95およびp99の待機時間を最小限に抑えた、Hedged Reads
  - レプリカの遅延を減らして、より新しいデータをユーザーに提供できる、 Simultaneous Indexing and Streaming Replication

MongoDB4.4の機能強化

- Resilience and Security
  - セカンダリレプリカのキャッシュを事前にウォームアップして、停止後または計画されたメンテナンス後のプライマリ選挙の影響を軽減できる、Mirrored Reads
  - 新しいレプリカを追加、また同期の遅れている nodeの回復など、スケールアウトをより簡単かつ迅速にできる、Resumable Initial Sync
  - MongoDB Atlasに接続するときに、既存の通常の temporary Amazon IAM認証情報を再利用し、クラウドネイティブのセキュリティを簡素化できる、 AWS IAM Authentication
  - TLS 1.3およびMongoDBへの50%高速なクライアント認証

ご清聴ありがとうございました