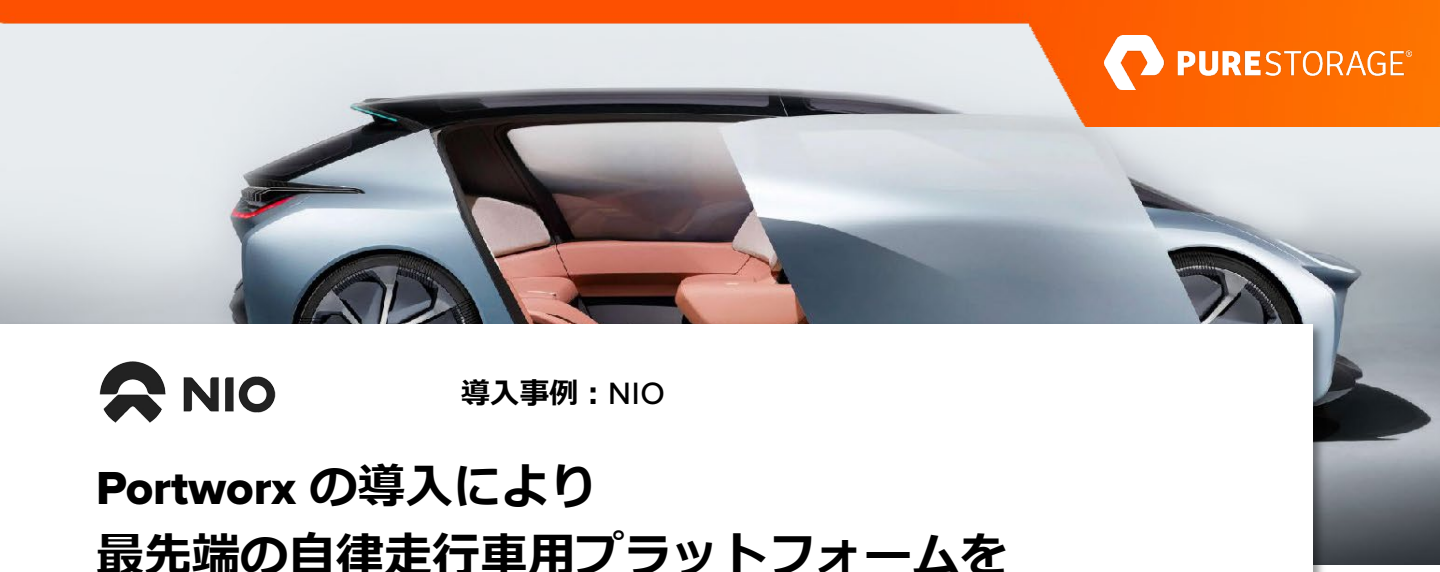

NIO

導入事例：NIO

Portworx の導入により 最先端の自律走行車用プラットフォームを コンテナベースで構築

課題

- 各自律走行車のネットワーク接続状況にあわせたコンピューティング／ストレージ・インフラの個別スケーリング
- 車両 1 台につき、1 日あたり最大 24 TB に達するデータのストレージおよび処理
- パフォーマンスの維持とワークロードの分離を両立させるには、ベアメタル上でコンテナを実行する必要があるが、Kubernetes および DC/OS でのデータ管理は不十分

ソリューション

- Portworx の導入により、Kubernetes および DC/OS で動作するアプリを対象に、柔軟で自動化されたデータ管理レイヤーを提供
- ストレージにコモディティ・サーバーを使用し、コンテナによる自動化のメリットを損なうことなく、ベアメタルと同等のパフォーマンスを達成
- Portworx による動的プロビジョニング、HA、スナップショット、バックアップ、データベースの暗号化、機械学習および深層学習のシナリオの実現

効果

- コンピューティングとストレージのインフラをそれぞれ迅速にスケーリングすることで、車両の生産拡大に柔軟に対応
- ストレージやデータ管理の主要なケイパビリティを損なうことなく、高密度、リソースの分離、コンテナのポータビリティといったメリットを最大化

主要テクノロジー

- コンテナ・ランタイム：Docker
- スケジューラ：Kubernetes、Mesosphere DC/OS
- ステートフル・サービス：Cassandra、Kafka、HDFS
- インフラ・プロバイダ：オンプレミス

電動自動車メーカー NIO は、Portworx を導入し、毎時発生する膨大なデータを管理・格納するためのコンテナ化されたインフラの構築および運用に成功しています。

https://www.nio.com/

※この資料では、NIO 自律走行車クラウド＆エンタープライズ・アーキテクチャ部門長 サティヤ・コマラ氏へのインタビュー取材の内容を抜粋してご紹介しています。

NIO
自律走行車クラウド＆
エンタープライズ・アーキテクチャ部門長
サティヤ・コマラ（Satya Komala）氏

NIO の事業はどのようなものですか？

NIO は、モビリティ企業として安全な自律走行電気自動車の開発に取り組んでいます。人々が自分の時間を取り戻し、よりよい人生を送れるようにすることを目標としています。現在の自動車は安全とはいえません。いったんハンドルを握れば、たとえ医師や教師、アーティスト、どんな人でも単なるドライバーにならざるを得ません。NIOは、自動車で安全に移動する方法を提供することで、より豊かな人生の実現を支援したいと考えています。

このような目標を実現するのが自律走行車です。現在 NIO では、レベル 2 とレベル 4 の 2 種類の車両の開発に取り組んでいます。ここでいうレベルは自律走行車特有の表現で、通常は 4 つのレベルが設けられています。レベル 1 と 2 では、安全についての責任はドライバーが担います。レベル 3、4、5 では、車両を製造する企業に責任が移ります。

レベル 1 は、自動ブレーキ・システム、緊急ブレーキ、レーン・アシストなどの基本的な機能を備えた車両です。現在、自律走行車の大半がこのレベルに属しています。レベル 2 では、現時点ではテスラ車に搭載されている高速道路でのオート・パイロット、ABS、自動クルーズ・コントロール、車線逸脱警報などの機能が含まれます。NIO では、レベル 2 を超え、安全確保の主導権がドライバーから車両に移るレベルに取り組み始めています。

レベル 3 は基本的に「アイズオフ」で、運転中に目を離すことのできるレベルとなります。テスラの現行車両の場合は、高速道路を走行中にハンドルから手を放すことはできません。道路から目を離すこともできません。車両が運転を支援しますが、依然としてドライバーが注意を払わなければなりません。一方、「アイズオフ」であるレベル 3 の車両では、ドライバーが道路に目を向けたり、ハンドルを保持したりする必要がなくなります。ただし、車両の判断により、自律走行からドライバー主体の運転への切り替えが随時発生するため、ドライバーは常に運転席にいる必要があり、睡眠をとることなどもできません。

レベル 4 は「アイズオフ、 マインドオフ」です。ドライバーが眠っていても、車両が自律的にフェイルセーフ機能を完了させます。重大な障害が発生した場合にも、ドライバーの関与なしで安全に車両が停止できるようになっていなければなりません。最上位のレベル 5 では、ハンドルやペダルすらなくなり、車内の人間は一切運転をしません。完全に車両だけで走行します。

NIO は現在、レベル 2 とレベル 4 の自律走行車に取り組んでおり、今年の上海モーターショーでレベル 2 を発表しました。来年の第 1 四半期にはお客様への提供を開始する予定です。ES8 という 7 人乗りの SUV です。「サウス・バイ・サウスウエスト」というイベントで発表したレベル 4 の自律走行車も開発中です。

> **“レベル 4 は「アイズオフ、マインドオフ」です。ドライバーが眠っていても、車両が自律的にフェイルセーフ機能を完了させます。重大な障害が発生した場合にも、ドライバーの関与なしで安全に車両が停止できるようになっていなければなりません。”**

レベル 4 の車両に関しては、車体、シャーシ、パワートレイン・システム、乗降システム、自律走行およびフェイルセーフ・システムの主要テクノロジーを全て自社で開発しています。「アイズオフ、マインドオフ」の車両においては、完全なフェイルセーフが求められます。私たちはその実現を目指して日々取り組んでいます。

どのような業務を担当されていますか？

テスラから NIO に転職して 14 か月。自律走行車のバックエンドを担当しています。ご想像のとおり、車両にはたくさんのコンピュータが内蔵されています。多くの電子制御装置が、通常はオンボード・コンピュータとその他のコンピューティング・デバイスに搭載されています。また、車両のネットワーキング機能を担い、車両で生成されたセンサー・データをクラウドに送信して処理するためのゲートウェイがあります。

私が担当しているのは、ゲートウェイを通じてデータが外部に送信された後のデータ処理全般です。データは全てデータ・プラットフォームに送信されます。NIO の自律走行エンジニアが、そのプラットフォーム上のデータに基づいて、車両の安全を維持するためのアルゴリズムと機械学習モデルを開発できるよう、エンドツーエンドのエクスペリエンスを提供するのが私の役割です。

コンテナをどのように利用していますか？

NIO が現在のソリューションにたどり着くまでにはちょっとしたエピソードがありました。私が以前在籍していたテスラでは、レベル 2 の車両を扱っており、クラウドを利用していたことから、NIO ではまずクラウド・プロバイダの評価から始めました。ところがすぐに、レベル 4 の車両では要件が全く異なることがわかりました。開発段階で、車両 1 台につき 1 日に 12〜24 テラバイトのデータが生成されます。10 台では 1 日 240 テラバイトになります。私の知る限りでは、それだけのデータをクラウドに保存する方法はありません。以前扱っていたものとは全く異なるレベルの問題でした。

データやアプリケーションではなく、インフラやシステムの検討から始める必要がありました。私は最適なインフラの構築について考えました。スタートアップ企業である NIO は、適切であると同時に、コスト効率のよい方法を選ばなければなりません。

そこで私たちは OpenStack を検討しました。ホワイトボックス・ハードウェアと、ハイパーバイザー上で動作する OpenStack VM を使用するアーキテクチャです。マイクロサービス向けにはコンテナを使用するのが通常の考え方なので、この時点で私は少し躊躇しました。「この方法は間違っているのではないだろうか？ハイパーバイザーは必要か？オーバーヘッドを許容すべきか？」

その後さらに調査を重ね、PoC を何度か実施し、NIO の革新性に見合うアプローチを採用すべきだという考えに至ります。膨大なビッグデータを扱う NIO のユースケース自体が既に革新的なものであり、そのために必要なものを構築すべきだと考えるようになったのです。そこでコンテナの採用という結論に至ります。NIO にはレガシー・アプリケーションがなかったこと、ハイパーバイザーのオーバーヘッドを回避したかったことも、コンテナ採用の背景となりました。仮想化を伴わないコンテナ中心の戦略が最も合理的な選択肢でした。

“仮想化は現在使用していません。Linux ベースのベアメタル・サーバーで Docker デーモンとコンテナを利用しています。”

仮想化は現在使用していません。Linux ベースのベアメタル・サーバーで Docker デーモンとコンテナを利用しています。インフラ全体が Super Micro および Lenovo を中心とするホワイトボックスをベースとしています。コンテナの管理にはオーケストレータを使用しており、Mesos と Kubernetes を導入しています。ステートフル・コンテナには Mesos、ステートレス・コンテナには Kubernetes の成熟度が高いと考えたためです。NIO では、Kubernetes の方向性を高く評価しており、ステートフル・コンテナのサポートについても徐々に向上していくと見込んでいます。目標の実現に向けて、今後もさらに合理的な方法を模索していきますが、現時点ではこの 2 つのオーケストレータをプラットフォーム上で実行しています。

コンテナでステートフル・サービスを実行するうえで、解決しなければならなかった課題について教えてください。

NIO のユースケースは、ごく一般的な IoT（モノのインターネット）のユースケースですが、異なる点があります。IoT から通常連想されるものとは反対に、少数のデバイスから大量のデータが送信されます。さらに、データの大部分は動画データです。

現在、公道を走る自律走行車には、車両のあちこちに複数のカメラが搭載されています。レベル 4 の車両では 8 台から 12 台のカメラが搭載されており、1080p の高画質の画像や動画が生成されます。これらのデータをプラットフォームに取り込み、処理する必要があります。車両ごとに 1 日あたり 12～24 テラバイトのデータが生成される状況を想像してみてください。

“Big Switch Networks というパートナーに巡り合い、ネットワーキング問題の解決に向けて協業することになりました。Big Switch のエンジニアとの連携を通じて CNI インターフェースのサポートを拡張し、現在はプラットフォームで安定したネットワーキング・ソリューションが利用できるようになっています。”

膨大な量のデータを処理するには、多くのステートフル・サービスを実行しなければなりません。NIO では Spark、Spark ML をはじめ、機械学習と深層学習のシナリオには TensorFlow、さらに Kafka、Kafka Streaming、Hadoop、Cassandra、MongoDB、ElasticSearch、MySQL を利用しています。

コンテナの構築を決定した当初から、いくつかの問題をクリアするためにパートナーとの協業が必要になることを認識していました。最も重大な問題は、ネットワーキングとストレージに関するものでした。

ネットワーキングについては、オーバーレイ・ネットワーキングに課題がありました。トラブルシューティングがほぼ不可能であることから、IP 空間の管理は大きな課題でした。コンテナを使用する場合には膨大な数の IP を保持することになり、管理しきれなくなるリスクがあります。そのため、CNI 標準に準拠した SDN ネットワークを用意し、よりフラットなネットワークとすることで、IP 空間とネットワーク空間の管理を自動化する必要がありました。幸いなことに、Big Switch Networks というパートナーに巡り合い、ネットワーキング問題の解決に向けて協業することになりました。Big Switch のエンジニアとの連携を通じて CNI インターフェースのサポートを拡張し、現在はプラットフォームで安定したネットワーキング・ソリューションが利用できるようになっています。

ネットワーキング問題が解決したところで、こんどはストレージの問題に取り組みました。そこで、NIO のデータ・フットプリントが巨大であるという問題に直面します。扱うデータ量が年間約 120 ペタバイトに達します。

“ネットワーキング問題が解決したところで、こんどはストレージの問題に取り組みました。そこで、NIO のデータ・フットプリントが巨大であるという問題に直面します。扱うデータ量が年間約 120 ペタバイトに達します。”

これだけのデータを処理するには、ニーズに応じて個別にスケーリング可能な汎用コンピューティングとストレージが必要でした。急成長する NIO では、常に新たな問題を解決していかなければなりません。ワークロードもまだ確定できず、さらに進化し続けることが予測され、将来的なニーズの見通しを立てにくいという問題がありました。

ただし、ストレージについては、SAN などのインフラと並行させるのではなく、NIO のインフラ上で直接実行するストレージ・ファブリックを導入することにしました。先に述べたように、NIO ではストレージを直接接続するホワイトボックス・ハードウェアを全面的に採用しています。Kubernetes と Mesos の両方と統合する仮想ボリュームの作成を可能にするストレージ・ファブリックが必要でした。コンテナを複数のノードに配置して、コンテナのアップ、ダウン、クリーンアップ、ボリュームの複製にあわせて仮想ボリュームを配分できるようにするためです。

この問題の一部は、高コストのストレージ・ベンダーが提供する、市場で入手可能なネットワーク接続ストレージ（NAS）、あるいは SAN を導入するという方法で解決できるかもしれません。しかし、SAN で 120 ペタバイトに及ぶデータを扱うことは困難であることを含め、私たちの条件に合う選択肢とはなりません。

そこで、ベンダーとの連携を通じて、私たちのニーズを実現するうえで既存のテクノロジーを活用できるかどうかを検討しました。複数の選択肢を調査し、GlusterFS や Ceph を使用する環境を試行しました。また、同様の製品でコンテナを対象としていないものについても検討しました。しかし、こういった取り組みは、次のような理由により、うまくいきませんでした。

Kubernetes 向けには、Gluster や Ceph を使用するための確実な方法が見つかりませんでした。レプリケーションの問題に加えて、コンテナやポッドのノード間のフェイルオーバー時の問題も多数見つかりました。”

第一に、GlusterFS と Ceph は運用が容易ではありません。これらのシステムの運用管理には多大な時間と労力を要します。第二に、Kubernetes や Mesos との統合の仕組みがなく、統合運用は極めて困難であることが判明します。コンテナベースのシステムを管理するということは、すなわち、ワークロードの運用のための膨大な数のコンテナを実行することを意味します。そのため、オーケストレータとストレージ・プロバイダを緊密に統合しなければなりません。

Gluster では、Mesos 用の REX-Ray プラグインがあることがわかりました。しかし、パフォーマンスやスケーリングの問題、管理上のオーバーヘッドが多く、信頼性に欠けるという結論に達します。また、GlusterFS と Ceph のファブリック自体が、サービスの実行にコンピューティング・ノードのリソースを大量に消費することから、NIO にとっては合理的な策ではありませんでした。

Kubernetes 向けには、Gluster や Ceph を使用するための確実な方法が見つかりませんでした。レプリケーションの問題に加えて、コンテナやポッドのノード間のフェイルオーバー時の問題も多数見つかりました。

このような評価の末にたどり着いたのが Portworx でした。

Portworx には、他のソリューションにはない安定性、効率性、使いやすさがありました。Portworx は、Mesos および Kubernetes との親和性が極めて高いことがわかりました。Kubernetes や Mesos のオーケストレータの一部として動作するため、シンプルで使いやすくなっています。

Mesosphere との緊密な統合も特長の 1 つです。NIO では Mesos の Universe パッケージを多数展開しており、それによってステートフル・ワークロードの管理が容易になっています。HDFS、Kafka、Elastic、Cassandra のいずれかでの運用する場合には特に効果が見込めます。これらのシステムを少人数のチームで管理・運用するのは容易なことではありませんが、Portworx のフレームワークを利用することで、負荷が大幅に軽減されます。

“Portworx には、他のソリューションにはない安定性、効率性、使いやすさがありました。Portworx は、Mesos および Kubernetes との親和性が極めて高いことがわかりました。Kubernetes や Mesos のオーケストレータの一部として動作するため、シンプルで使いやすくなっています。”

NIO では、既にプラットフォームの最初のバージョンを立ち上げ、車両向けに運用しています。わずか 15 名のエンジニアで構成するチームで、システムを担当しているのはそのうちの 3 名だけです。ビルトインのスケジューラ統合や Portworx によるパッケージの操作が直感的で使いやすいため、このように少人数での運用管理が可能になっています。この点は、私が特に気に入っている Portworx の特長の 1 つです。

サポート体制も充実しています。Portworx エンジニアリング部門の VP が私たちとともにトラブルシューティングを行なってくれましたが、そのような手厚いサポートは他では経験したことがなく、感銘を受けました。Portworx のサポートがなければ、NIO のプロジェクトがここまで順調に進むことはなかったでしょう。顧客の成功に重点を置くという Portworx チームの姿勢はすばらしく、ここ数週間のサポートはまさに驚くべきものでした。

多くのオープンソース・コンポーネントを採用されていますが、オープンソースのコンポーネントとベンダー・ソリューションをどのように使い分けるのがよいでしょうか？

リスクを理解したうえで行動することが重要だと考えています。オープンソースの製品は多数存在し、コミュニティによるサポートも充実しています。ただし、スタックの中には決して障害を発生させてはならないものがあります。障害による影響が甚大で取り返しのつかないものについては、方向性を決める時点で慎重な検討が必要です。なかでもストレージは特に重要なコンポーネントです。オープンソースという選択肢もあるかもしれませんが、NIO のユースケースに十分な安定性を提供できるとは考えられませんでした。私たちは、リスクを回避し、厳密な検証を通じて高度な安定性が保証されているベンダーのソリューションに委ねるほう合理的だと判断しました。また、クラウドネイティブなアプリケーションの標準的なインターフェースのための CNI や CSI などのプロジェクトがあり、切り替えが必要になった場合のコスト抑制を可能にしています。

“スタックの中には決して障害を発生させてはならないものがあります。障害によって甚大な影響が出るものに関しては、方向性を決める時点で慎重な検討が必要です。なかでもストレージは特に重要なコンポーネントです。”

ステートフル・コンテナの実運用について、何かアドバイスはありますか？

まず、レガシー・アプリケーションによる技術的負債がないことが重要です。 技術的負債が大きすぎる場合には、別のアプリケーションのコンテナ化を検討したほうがよくなります。コンテナのみ、またはコンテナ優先のアプローチを取る場合には、レガシー・アプリケーションをコンテナにパッケージ化する方法を考える必要があります。

コンテナをベースとした開発環境への移行は、開発者にとっては新しい試みになりますが、いったん理解すれば、優れた開発エクスペリエンスが期待できます。また、ネットワーキングとストレージの問題は、発生した場合の負担が非常に大きくなるため、解決しておく必要があります。

適切なパートナー、ベンダーを選択し、ストレージとネットワーキング周りは万全にしておくことです。この 2 つの重要領域については、お薦めできるソリューションはオープンソースには存在しません。

Portworx について

ピュア・ストレージの Portworx は、Kubernetes のための包括的なデータ・サービス・プラットフォームとして、永続ストレージ、データ保護、ディザスタ・リカバリ、データ移行、容量管理のための Kubernetes ネイティブな統合ソリューションを提供します。単一のデータ管理レイヤーが、実行環境を問わず、データベースをはじめとするあらゆるステートフル・サービスに対応し、複数のクラウドやオンプレミスのハイブリッド環境におけるコンテナ化されたアプリケーションの迅速なデプロイメントを可能にします。Portworx は、多くのエンタープライズにおいて、デプロイメントのコストおよび工数の削減に貢献しています。

ご相談・お問い合わせをお待ちしております。

PURESTORAGE®

ピュア・ストレージ・ジャパン株式会社
お問い合わせ：03-4563-7443（代表）
https://www.purestorage.com/jp/contact.html