

Operating System Support for Safe and Efficient Auxiliary Execution

山下 恭平

概要：本稿は USENIX, OSDI'22 に掲載されている論文「Operating System Support for Safe and Efficient Auxiliary Execution」^[1] の内容についてまとめたものである。近年のアプリケーションは様々な補助タスクが実行されている。補助タスクとは、アプリケーションが自身のメンテナンス、自己管理を行うタスクのことである。これらのタスクは、アプリケーションのアドレス空間で実行することで、高い観測性と制御性を得るが、安全性と性能の問題が発生する。また、補助タスクを別のプロセスで実行すると、分離性は高いが、観測性と制御性が劣る。本稿では、この問題を解決するために、補助タスクに対するサポートとして、orbit と呼ばれる OS の抽象化機能を提案する。

1. はじめに

運用されているアプリケーションはその実行状況を調査し、最適化し、デバッグし、制御するために、頻繁にメンテナンスを行う必要がある。かつてはメンテナンスはアプリケーションの管理者が手動で行なっていたが、現在では多くのアプリケーションは、自身でメンテナンスを行うための補助タスクが行われている。例えば MySQL では、デッドロックを検出するとロールバックを行う機能が存在する。^[2] そのため補助タスクはアプリケーションの信頼性や観測性に大きく影響する。既存の OS に搭載されているプロセスやスレッドといった抽象化機能はメインタスクの実行に適した設計がされており、補助タスクの実行には適していない。そのため、開発者は、分離は強いが観測と制御が非常に限定される（別プロセス）か、観測と制御は強いが分離はほとんどできない（スレッド）かのどちらかを選ばざるを得ない。この問題を解決するために、OS の補助タスクに対するサポートを提供する orbit 抽象化を提案する。

orbit タスクは、協力的な分離を提供する。同時に、状態同期機能によりメインタスクを観測することも可能である。orbit のプロトタイプは Linux kernel 5.4.91 に実装された。orbit の評価を行うために MySQL, Apache を含む 6 つの大規模アプリケーションから、7 つの補助タスクを抽出し、orbit に移行することに成功した。また、全てのケースでアプリケーションはフォールトから保護されることがわかった。分離のコストを測定した所、orbit バージョンアプリケーションでは、中央値で 3.3 % のオーバーヘッドが発生した。

本稿では 2 章でこの研究が行われる背景について述べ、3 章で orbit の詳細な説明を行う、4 章で orbit の性能評価を行い、最後の 5 章では全体のまとめる。

2. 研究の背景

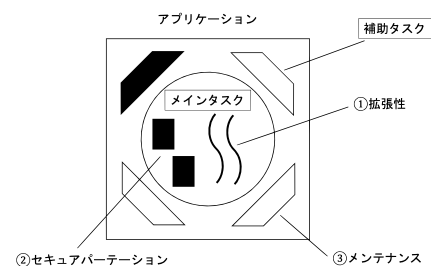


図 1 アプリケーション構成

アプリケーションはメインタスクと補助タスクの 2 つの論理領域に分けることができる。図 1 はアプリケーションの構成を示したものであり、メインタスクの周辺に補助タスクが存在することがわかる。(1) から (3) については後から説明を行う。補助タスクはメインタスクのサポートを行うために、メインタスクを観測し、制御する必要がある。この時、補助タスクのバグによってメインタスクに影響が出ないようにするため、メインタスクと補助タスクの隔離を行う必要がある。つまり、補助タスクには、観測性、制御性、隔離が必要である。

2.1 タスクの実行方法

既存の OS では補助タスクは主に 2 種類の実行方法をとる。1 つ目は、メインプログラムと同一のアドレス空間に配置され、メインタスクの関数、スレッドとして実行される方法だ。この方法であれば、補助タスクはメインタスクの観測、制御が容易に行うことができる。しかし、補助タスクによってメインタスクが不必要なブロックが起こる可能性や、補

助タスクにバグが存在した場合、メインタスクにまで影響が出るため、隔離について不十分である。2つ目はメインタスクと補助タスクを別プロセスで実行する方法だ。この方法であれば、メインタスクと補助タスクは異なるアドレス空間に存在するため、隔離が十分に行えている。しかし、メインタスクの観測と制御が困難になる。プロセスやスレッドといった機能は1章で述べた通り、補助タスクの実行には適していないことがわかった。

2.2 タスク実行の保護

タスク実行を保護する OS のサポートは既に多く研究、提案されている。しかし、それらは異なる2つの目的のために設計されている。

1つはアプリケーションの拡張性のためである(図1,(1))。具体的には SFI^[3] という技術がある。SFI とは、アプリケーションバイナリを書き換えることで、アプリケーション内の信頼できないコードのメモリアクセスを制限するソフトウェア隔離技術ことである。この技術はブラウザの拡張機能などの第三者が作成したプログラムをアプリケーション内で動作させるときに有効である。

2つ目は、セキュアなパーティショニングを提供するもの(図1,(2))。具体的には Wedge^[4] や lwc^[5] といったものがある。これにより、アプリケーションが侵害された場合に、メインタスクの機密な手続きを守ることができる。

しかし、これらの仕組みはアプリケーションのメンテナンス(図1,(3))にとっては不十分である。その理由として、まず、補助タスクはアプリケーションの開発者と同一人物、機関によって開発されているため、信頼できるものである。また、補助タスクはメインタスクと対話的であり、メインタスクの状態を常に監視し、場合によっては変更を加える必要があるからだ。次の章では補助タスクのサポートに求められるものについて述べる。

2.3 補助タスクの安全性と性能

1章で述べたように、メインタスクと補助タスクが同一のメモリ空間に存在する場合、補助タスクのバグにより、無効なメモリにアクセスをした場合アプリケーション全体がクラッシュすることが考えられる。メインタスクと補助タスクを分離した場合、補助タスクのバグによってアプリケーション全体がクラッシュする事態は防ぐことができる。しかし、分離された補助タスクによってメインタスクの性能が大きく低下する場合がある。MySQL のデッドロック検出タスクを実行した状態で、性能を測定したところ、スループットが最大で 79.5%低下することが報告されている。^[6] このことから、補助タスクはアプリケーションの性能を維持、向上させる機能を持っているのにも関わらず、逆にアプリケーションに害を与える可能性が存在する。

補助タスクの安全性と性能に関する懸念に対処するため

に、Fork-based Execution Model と Sandbox-based Execution Model という2つの代替案がある。

Fork-based Execution Model

Sandbox-based Execution Model

3. orbit について

4. orbit の性能評価

5. おわりに

参考文献

- [1] Yuzhuo Jing, Peng Huang. Operating System Support for Safe and Efficient Auxiliary Execution, 16th USENIX Symposium on Operating Systems Design and Implementation, page 633 - 648 (2022).
- [2] Oracle. MySQL's deadlock detection. <https://dev.mysql.com/doc/refman/8.0/en/innodb-deadlock-detection.html> (11/25/2022)
- [3] R. Wahbe, S. Lucco, T. E. Anderson, and S. L. Graham. : Efficient software-based fault isolation. In Proceedings of the Fourteenth ACM Symposium on Operating Systems Principles, SOSP '93, page 203 - 216, (1993).
- [4] A. Bittau, P. Marchenko, M. Handley, and B. Karp. Wedge: Splitting applications into reduced-privilege compartments. In Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, NSDI '08, page 309 - 322, (2008).
- [5] J. Litton, A. Vahldiek-Oberwagner, E. Elnikety, D. Garg, B. Bhattacharjee, and P. Druschel. Light-weight contexts: An OS abstraction for safety and performance. In Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, OSDI '16, page 49-64, (2016).
- [6] InnoDB deadlock detection is CPU intensive with many locks on a single row. <https://bugs.mysql.com/bug.php?id=49047>. (11/25/2022)