Cellソフトウェア開発環境

Cell Software Development Environment

大澤　諭　　内川　貴幸　　高野　秀隆
■OSAWA Satoshi　■UCHIKAWA Takayuki　■TAKANO Hidetaka

東芝は，Cellソフトウェア開発環境として独自に，Eclipse統合開発環境，PPE（PowerPC Processor Element）/SPE（Synergistic Processor Element）シームレスデバッガ，及びパフォーマンスモニタを開発した。シームレスデバッガ Cell Broadband Engine-GNU Debugger（CBE-GDB）は，GNUのgdbをベースにし，PPE/SPEのプログラムを同時にデバッグできるような拡張を行った。Eclipseに対しては，C/C++開発ツール（CDT：C/C++ Development Tool）をCBEに対応した拡張プラグイン（CBE-CDT）を開発し，CBE-GDBを使ってPPE/SPEのプログラムをGUI（Graphical User Interface）でデバッグできるようにした。パフォーマンスモニタは，CBE内部のパフォーマンス機能を使って，CBEの命令発行状況や内部バスのバンド幅などを測定できるようにした。

Toshiba has developed an Eclipse-based integrated development environment, a PPE/SPE (PowerPC Processor Element/Synergistic Processor Element) seamless debugger, and a performance monitor to make up our Cell software development environment. The debugger, named CBE-GDB (Cell Broadband Engine-GNU debugger), is based on the GNU debugger and has been extended to enable the debugging of PPE and SPE programs simultaneously. We have developed a Cell-compatible C/C++ development tool (CDT) named CBE-CDT as an extended plug-in for Eclipse, which makes it possible to use a graphical user interface to debug PPE/SPE programs with CBE-GDB. The performance monitor uses performance monitoring functions inside Cell to enable the user to measure the number of instructions issued, the bandwidth of an internal bus, etc.

1　まえがき

東芝は，Cell Broadband Engine（CBE）のLinux(注1)アプリケーションを開発するユーザーの生産性向上を目指して，Cellソフトウェア開発環境を開発した。CBEのソフトウェアは，PPE（PowerPC Processor Element）上のメインプログラムと，SPE（Synergistic Processor Element）上のプログラムがそれぞれ協調しながら動作する。PPE/SPEそれぞれのプログラムは，C/C++言語で記述できる。プログラマは，C/C++言語拡張仕様で定義された組込み関数（intrinsic）を使うことで，SPEのSIMD（Single Instruction Multiple Data）命令やSPE内のハードウェア制御命令をC言語で記述できる。一方，デバッグ作業では，PPE/SPEのプログラムを個々のデバッガでデバッグしようとすると，デバッガの起動や設定がめんどうになるだけでなく，一つのプログラムのブレークに同期して他のプログラムを停止させることが難しいという問題がある。

当社は，このようなPPE/SPEのソフトウェアを容易に開発できるソフトウェア開発環境を開発した（**図1**）。

Eclipse統合開発環境は，Eclipseプラットフォームと，GNUツールチェイン（コンパイラgccとデバッガgdb）を利用したGUI統合開発環境であり，最近の組込みソフトウェアのトレンドにもなっている。

コンパイラは，PPEプログラム用のppu-gccと，SPEプログラム用のspu-gccがある。これらは，標準C/C++言語ライブラリ（glibc）とバイナリユーティリティ（binutils）を含め，（株）ソニー・コンピュータエンタテインメント，ソニー（株），IBM Corporation，及び東芝で共同開発したものを利用している。

デバッガは，PPE/SPEのプログラムを同時にシームレスにデバッグできるようにするため，当社が独自に拡張した。

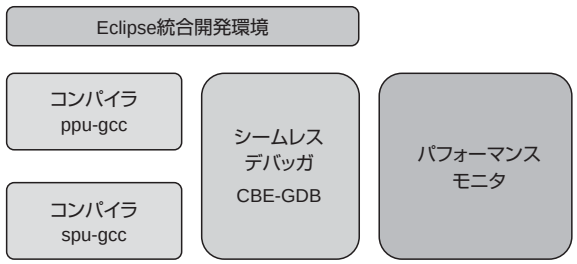

図1．Cellソフトウェア開発環境のラインナップ― Eclipse統合開発環境，コンパイラ，デバッガ，パフォーマンスモニタで構成されている。
Lineup of Cell software development environment

（注1）　Linuxは，Linus Torvalds氏の米国及びその他の国における登録商標。

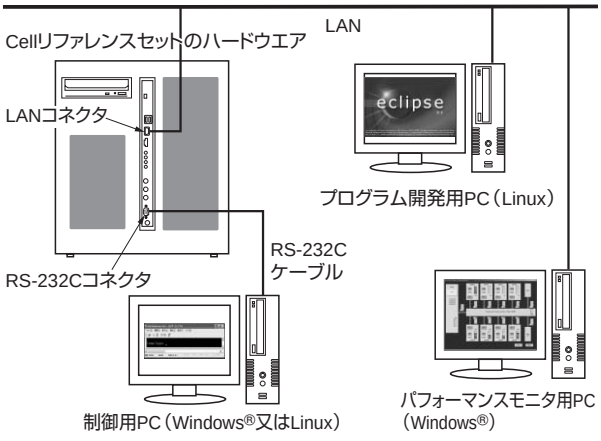

図2．機器構成 — Cellリファレンスセットのハードウェアと PC を図のように接続する。制御用 PC のシリアルコンソールには，プログラム開発用 PC あるいはパフォーマンスモニタ用 PC が使用できる。

Configuration of machines

プログラムのボトルネックの解析とチューニングを行うため，パフォーマンスモニタを当社が独自に開発した。これは，CBE内部のパフォーマンスモニタ機能を利用することで，専用ハードウェアを使用せずにCBE内部の動作情報（命令発行状況，ストール回数，バスバンド幅など）を詳細にモニタすることができる。

また，Cellソフトウェア開発環境を使用するときの機器構成を**図2**に示す。

2　Eclipse統合開発環境とシームレスデバッガ

Eclipseは，オープンソースコミュニティ（http://www.eclipse.org/）によって作られている統合開発環境で，プラグインによる機能拡張ができるという特長がある。ユーザーは，CDTプラグインとGNUツールチェインを追加することで，C/C++プログラムのコーディング，コンパイル，デバッグ，及びソースコードのリビジョン管理を行うことができる。なお，Eclipse，CDT，GNUツールチェインは，すべてオープンソースである。

当社は，CBEのPPE/SPE非対称マルチコア構成にシームレスに対応させるため，独自の拡張プラグインCBE-CDTとCBE-GDBを開発した。前者はCDT（v3.0.0）を，後者はGNUのgdb（v6.3）をベースにしている。

まず，PPE/SPEプログラムが，Eclipse統合開発環境とデバッガの上で，どのようにシームレスにデバッグできるかを述べる。Eclipseのデバッグ用画面（デバッグパースペクティブ）を**図3**に示す。

左上のデバッグビューには，デバッグ対象プログラムで生成されたPPEスレッドとSPEスレッドの状態が表示される。左側では，プログラムがPPEとSPEで各1個ずつ動いていて，SPEでブレークしている状態である。左下のエディタビューには，SPEプログラムの各ソースコードが表示される。右上のレジスタビューには，SPEのレジスタの値が表示される。右下の逆アセンブルビューには，SPEの逆アセンブルコードが表示される。

プログラムがPPEのブレークポイントで停止したときは，図3右側のように，PPEのソースコード，レジスタ値，及び逆アセンブルコードの表示に切り替わるようにした。再びSPEのブレークポイントで停止したときは，SPEのソースコード，レジスタ値，及び逆アセンブルコードの表示に切り替わるようにした。また，デバッグビューでスレッドを切り替えたときには，エディタビュー，レジスタビュー，逆アセンブルビューに表示するものを，選択したスレッドに対応したものに自動的に切り替わるようにした。

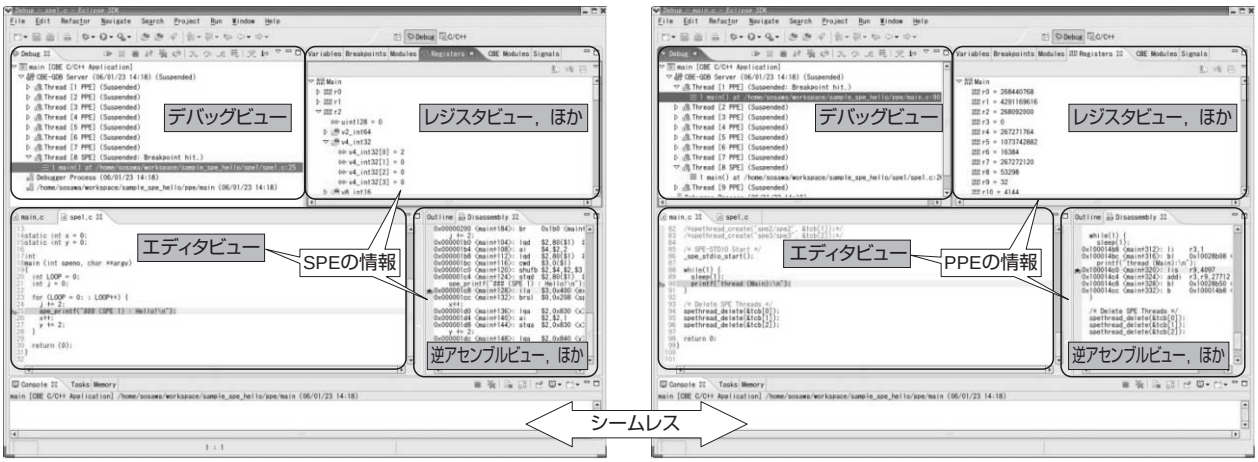

図3．Eclipseのデバッグ用画面 — SPEでブレークしたときはSPEの情報を表示し（左側），PPEでブレークしたときはPPEの情報を表示する（右側）。

Screen shot of Eclipse for debugging

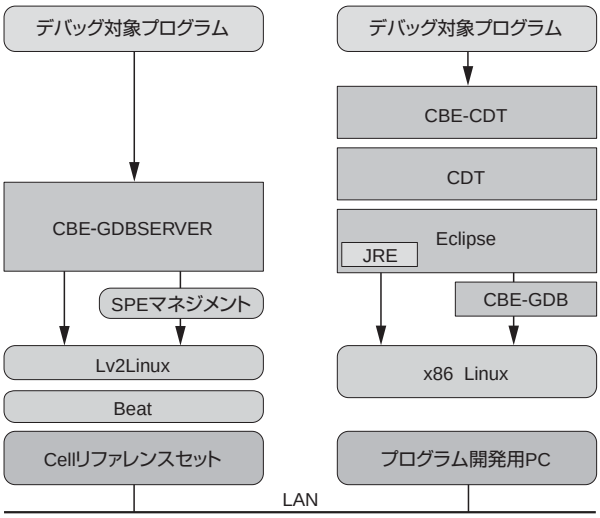

図4．デバッグ時のソフトウェア構成 ― Cellリファレンスセット上でCBE-GDBSERVERを動かし，プログラム開発用PC上のEclipseでリモートデバッグする。

Software structure for debugging

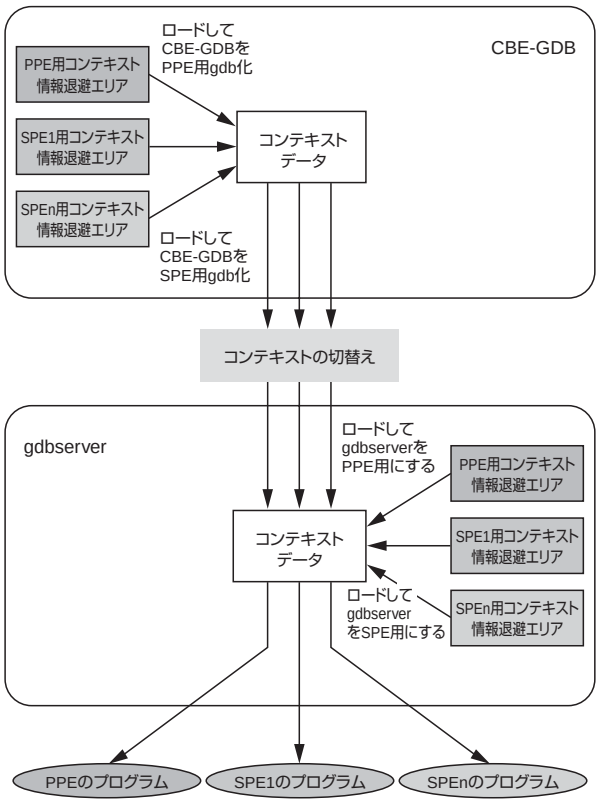

図5．gdb/gdbserverに対して行った変更内容 ― 非対称マルチコアプロセッサ構成に対応できるようにするため，各PPE/SPEのデバッグコンテキストを切り替える方式を採用した。

Modified gdb/gdbserver

従来，マルチコアのプログラムをデバッグするときには複数のデバッガを立ち上げる必要があったが，このように，一つのデバッガでPPEとSPEのプログラムをシームレスにデバッグすることができるようにした。また，CBE-CDTの操作感をオリジナルのCDTと変わらないものにし，ユーザーに特別な習熟を必要とさせないように配慮した。

次に，デバッグ時のソフトウェア構成を説明する（**図4**）。

デバッガは，ホスト側でEclipseからのコマンドを受けるCBE-GDBと，実機側でデバッグ対象プログラムの実行を制御するCBE-GDBSERVERから成る。デバッグ作業は，この二つを連携させてリモートでデバッグを行う“クロスデバッグ手法”で行うことになる。

Cellリファレンスセット側は，基本ソフトウェア（Beat，Lv2Linux，SPEマネジメント）の上でCBE-GDBSERVERを起動する。このとき，デバッグ対象プログラムを指定する。

また，プログラム開発用PC（パソコン）側は，FedoraCore4などのx86 Linuxの上でEclipse（v3.1.0）を起動する。CBE-GDB，CDT，CBE-CDTは，Eclipseから起動される。なお，Eclipseを動かすには，JRE（Java Runtime Environment）も必要である。

次に，シームレスデバッグを実現するために，オープンソースのgdb/gdbserverとCDTに対して行った変更内容について述べる。

gdbは本来，単一アーキテクチャプロセッサのアプリケーションをデバッグするように作られている。一つのCBE-GDBがPPE/SPE非対称マルチコアアーキテクチャに対応するようにするために，各PPE/SPEスレッドをデバッグするのに必要な情報（コンテキスト）を切り替える方式を採用した（**図5**）。例えば，PPEスレッドで動き始めたプログラムがSPEスレッドでブレークしたときは，そのPPEスレッドのコンテキストを退避させ，ブレークしたSPEスレッドのコンテキストをロードしてSPEのgdbとして動作させる。

またgdbのCBE対応として，CBE-GDBにSPE内のMFC（Memory Flow Controller）のレジスタを参照・設定するコマンドを追加した。CBE-GDBとCBE-GDBSERVER間のプロトコルには，前述の追加コマンドに対するものや，SPEステータスを取得するものなどを追加した。更にCBE-GDBSERVERには，SPEのプログラムを制御する仕組みを追加するとともに，前述の追加コマンドの対応を行った。また，CDTからgdbを制御するとき，gdb/マシン非依存インタフェース（MI）を使用するが，CBE向けに拡張する必要があるコマンド（ブレークポイント関連，データアクセス関連，MFC関連など）に対して，新たにCBE用のgdbコマンドを追加した。

オリジナルのCDTが持つ機能には，エディタ（文法強調，コード補完など），デバッガ（gdbのGUI化），ランチャー（プログラムやデバッガの起動），パーサー，インデクサ，検索，Makefile生成，及びCVS（Concurrent Versions System）を

利用したソースコード構成管理機能などがある。CBE-CDTでは，このうちデバッガ機能のスレッド表示，ブレークポイント設定，レジスタ参照，メモリ参照，逆アセンブリ，MFC参照機能などに対して，それぞれPPEとSPEを区別できるようにするとともに，CBE-GDBを起動できるようランチャー機能を拡張した。これらのCBE対応は，オリジナルEclipse/CDTの機能をオーバライド（上書き）することで，オリジナルを改造することなく実現した。

3　パフォーマンスモニタ

CBEにはハードウェアの動作情報を記録するパフォーマンスモニタ機能があり，あらかじめハードウェア内に準備された多数の測定項目から，1回の測定につき4～8個の測定項目を最短10サイクルごとにCBE内部のバッファ又は外部メモリに連続して記録することができる。

従来のパフォーマンス測定では，高価な専用ハードウェアを利用して高精度なデータ測定を実施するか，ソフトウェアにより低精度のデータ測定を実施するのが実情であった。そこでCBEではチップ内部に高機能なパフォーマンスモニタ機能を実装し，ソフトウェアから利用可能としたことにより，専用ハードウェアを用いずにソフトウェアからの高精度なデータ解析を可能としている。

CBEのパフォーマンスモニタのソフトウェアは，この機能の簡易な利用方法から，ユーザー解析が容易になる形での性能データの表示方法までをまとめて，効率のよいチューニング環境を提供している。

まずハードウェア全体の概要をリアルタイムに表示する形式（パフォーマンスメーター）を**図6**に示す。

このグラフでは指定した周期ごとに，各SPEの動作状況（2命令同時実行，1命令実行，チャネルストール（DMA（Direct Memory Access）転送待ちによるストール），及びブランチミス（分岐ミスによるストール），それぞれのサイクル数の割合），PPEの命令実行状況，内部バスとSPE，PPE，XIO(注2)，FlexIO(注3)間のデータ転送量が表示される。この機能は，パフォーマンスモニタ用PCからネットワークを介してデータを取得し表示をするため，更新周期はms単位となり全体の挙動把握に使用する。

次に，詳細なデータの記録を行うための測定及びグラフ表示形式について述べる。SPEの2命令同時実行，1命令実行，チャネルストール，ブランチミスの測定例を**図7**に示す。

このグラフの作成には，パフォーマンスモニタ用PCでのGUIツールを利用する。この開発の特徴として，ハードウェア内に準備された多数の測定項目から，ソフトウェアのチューニングに重要な項目を抽出してユーザーに提供していることが挙げられる。実際にハードウェアが提供している測定項目はハードウェアに近い情報であり，測定のための情報設定も多岐に及ぶため，ソフトウェア開発者が適切にそれらを理解して使用するには若干の困難があるためである。

まず，測定項目の分類では，ハードウェアの機能に応じて四つの測定グループに大別する。各測定グループ内では更に細かな複数の測定指標を設定し，各測定指標内には1回で測定可能な4～8個の測定項目を設定している。それらの

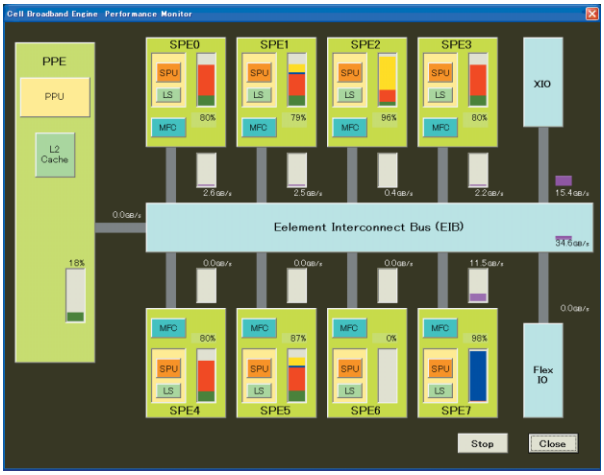

図6．パフォーマンスメーター ― ハードウェア全体の概要をリアルタイムに表示する。
Performance meter

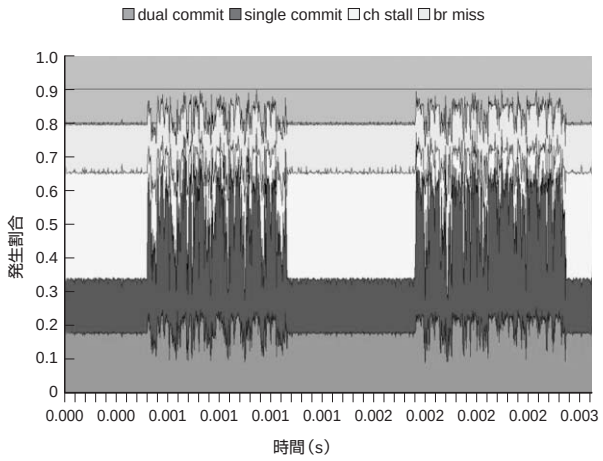

図7．性能データの例 ― SPEの2命令同時実行，1命令実行，チャネルストール，ブランチミスの測定例である。
Example of performance data

(注2)，(注3)　XIO，FlexIOは，Rambus社の登録商標。

表1．測定グループ
Measurement groups

No.	データの種類
0	SPE コア関連
1	チャネルストール関連
2	データバンド幅関連
3	バスコマンド関連

表2．測定指標の例
Examples of measurement indexes

No.	データの種類
0	SPE 命令実行状況
1	SPE ブランチ実行状況
2	SPE コアサイクル数消費の内訳1
:	:

表3．測定項目の例
Examples of measurement items

No.	データの種類	グラフでのラベル名称
0	2命令同時実行回数（%）	dual commit
1	1命令実行回数（%）	single commit
2	Pipeline0 命令実行回数（%）	pipe0 commit
3	Pipeline1 命令実行回数（%）	pipe1 commit

一部を**表1**～**表3**に示す。更に，測定項目に応じた設定情報の準備や取得したデータの加工と単位付けをしたグラフ表示をすることにより，ユーザーは主要測定項目を選択するだけで，測定からグラフ表示までを行うことができ，ユーザーの利便性が大幅に向上している。

パフォーマンスモニタの構成を**図8**に示す。GUIツールはパフォーマンスモニタ用PC（OS：Windows®(注4) XP）のExcelにて動作し，Cellリファレンスセット上のパフォーマンスモニタの制御，データ転送を行う。パフォーマンスモニタは，各OS（Beat，Lv2Linux，SPEマネージメント）を介してCBEのパフォーマンモニタ機能の設定やデータ収集を行い，ファイル保存とGUIツールへのデータ転送を行う。

なお，この機能はCBEの動作状態を測定するため，測定対象プログラムに加え，測定期間中に動作したOSや他のプログラムの動作情報も併せて記録するので注意が必要である。

このツールの開発により，専用ハードウェアを使用せずに，標準スペックのWindows® PCを利用して高精度なCBEの動作情報の記録を実現した。このツールは実際のAVアプリケーション開発に利用されており，事前に準備した測定項目を利用して，短時間でデータ解析から性能改善までできる環境を提供した。

（注4） Windowsは，米国Microsoft Corporationの米国及びその他の国における登録商標。

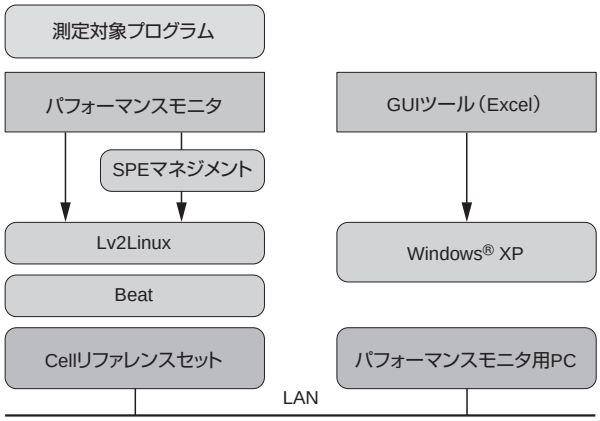

図8．パフォーマンスモニタの構成 ― パフォーマンスモニタが動作するハードウェア及びソフトウェアの構成である。
Structure of performance monitor

4　あとがき

Cellソフトウェア開発環境として，Eclipse統合開発環境，PPE/SPEシームレスデバッガ，及びパフォーマンスモニタを開発した。

今後の展開としては，パフォーマンスモニタとデバッガの連携や，コードカバレッジツールの開発などを考えている。また，共通プラットフォームに対応したソフトウェア開発環境の開発を進めていくとともに，オープンソースコミュニティへの貢献もしていきたい。

大澤　諭　OSAWA Satoshi
セミコンダクター社 ブロードバンドシステムLSI事業推進部 ブロードバンドシステムLSI開発センター主務。Cellソフトウェア開発環境の開発・設計に従事。
Broadband System LSI Div.

内川　貴幸　UCHIKAWA Takayuki
セミコンダクター社 ブロードバンドシステムLSI事業推進部 ブロードバンドシステムLSI開発センター。Cellソフトウェア開発環境の開発・設計に従事。
Broadband System LSI Div.

高野　秀隆　TAKANO Hidetaka
東芝情報システム（株）エンベデッドプラットフォーム・ソリューション事業部第二ソリューションセンター主任。開発環境及びリアルタイムOSの開発・設計に従事。
Toshiba Information Systems (Japan) Corp.