Hewlett Packard Enterprise

Reference Architecture

HPE Reference Configuration for Intel Arria 10 GX FPGA with Swarm64 Data Accelerator on HPE ProLiant DL380 Gen10

World's First Enterprise-Class Intel® FPGA on HPE ProLiant DL using Business Intelligence (BI) Data Analytics workload

Contents

Executive summary

The Intel® Arria® 10 GX FPGA + Swarm64 DA + HPE ProLiant DL380 is an engineered solution to accelerate analytical and hybrid analytical transactional processing in PostgreSQL with the help of the Intel Programmable Accelerator Card.

The Field Programmable Gate Array (FPGA) accelerator card for data centers offers both inline and lookaside acceleration. It provides the performance and versatility of FPGA acceleration and is one of several platforms supported by the Acceleration Stack for Intel Xeon® CPU with FPGAs. The card can be deployed in a variety of HPE servers with its full-height form factor, low-power dissipation, and passive heat sink.

With the Intel Arria 10 GX FPGA + Swarm64 DA on PostgreSQL solution, HPE adopts and provides to customers an exciting new platform for deploying database solutions with substantially improved performance at much better TCO with memory and/or storage bounded systems. This solution brings the high-performance, object-relational database engine to the enterprise Linux® ecosystem.

HPE provides the first-of-its-kind Intel FPGA solution to accelerate analytics database performance on HPE ProLiant servers with these benefits:

- **Up to 2.7x faster performance** using the Business Intelligence/Data Analytics queries
- **Up to 41% lower TCO-performance ratio** using the Business Intelligence/Data Analytics queries

This solution is ideal for:

- Customers who are looking to improve cost/query processing on general purpose and Entry/Mid-level Intel Arria 10 GX FPGA + Swarm64 DA on PostgreSQL Database deployments
- Customers running some versions of PostgreSQL and seeking better query processing performance
- Database customers who are looking for accelerated analytics performance and TCO-performance solutions and benefits

Target audience: This Hewlett Packard Enterprise Reference Configuration white paper is designed for IT professionals who use, program, manage, or administer large databases that require high performance and better TCO-performance advantage. Specifically, this information is intended for those who evaluate, recommend, or design new IT high-performance architectures.

Document purpose: The purpose of this document is to describe a data accelerator FPGA Reference Configuration, highlighting recognizable benefits to technical audiences.

Solution overview

This solution is based upon Intel Arria 10 GX FPGA with Swarm64 Data Accelerator (DA) using PostgreSQL. The underlying hardware is the HPE ProLiant DL380 Gen10 server. An overview of each of these components is provided below.

Intel Arria 10 GX FPGA

The Intel Arria 10 GX FPGA is the highest performance FPGA and SoC at 20 nm. This FPGA implements publicly-available OpenCore designs. Intel Arria 10 GX FPGA and SoCs feature the industry's only hard floating-point digital signal processing (DSP) blocks with speeds up to 1.5 tera floating-point operations per second (TFLOPS). The Intel Arria 10 GX FPGA and SoCs are ideal for a broad array of applications such as communications, data center, and military, broadcast, automotive, and other end markets.

Intel Arria 10 GX FPGA with Swarm64 DA

The Intel Arria 10 GX FPGA with Swarm64 DA plugs into PostgreSQL as a PostgreSQL extension and uses the Foreign Data Wrapper as a storage interface for table data and indexes. The Intel Arria 10 GX FPGA with Swarm64 DA uses both the CPU and the Intel Programmable Accelerator Card (Intel PAC) to share the workload during data ingestion and analytics.

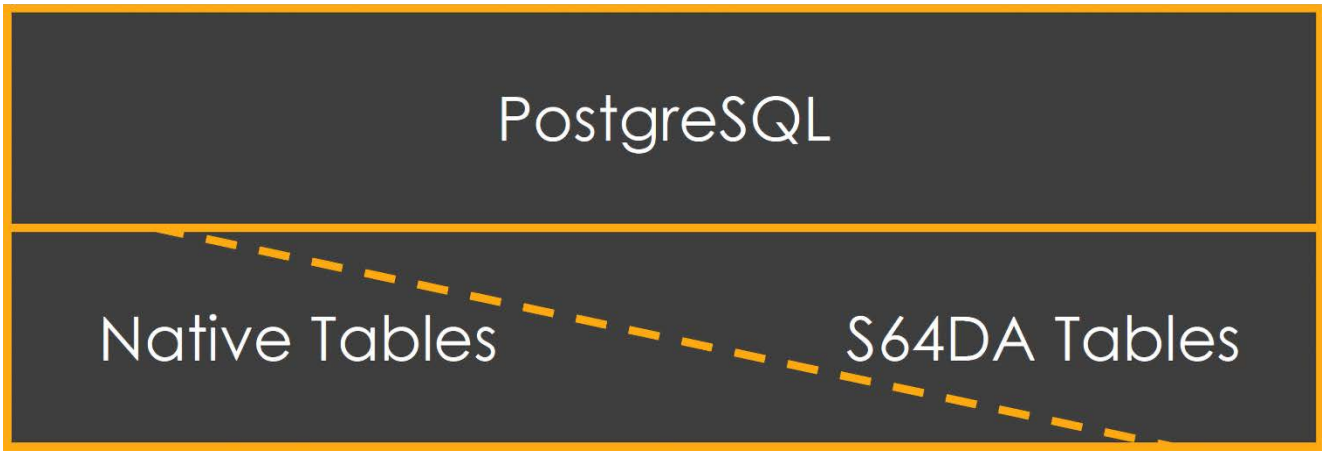

Figure 1. Intel Arria 10 GX FPGA with Swarm64 DA supports all of the database's functionality via the PostgreSQL native table format or a large sub-set via the Intel Arria 10 GX FPGA with Swarm64 DA tables, created as "Foreign Tables".

Optimized Columns and when to use them

The Intel Arria 10 GX FPGA with Swarm64 DA employs a proprietary mechanism, Optimized Columns, to arrange the data for FPGA and database processing. Optimized Columns allow the Intel Arria 10 GX FPGA with Swarm64 DA to very efficiently process frequent values or value ranges in any of these columns, in any order. Reading through these ranges is also highly optimized for I/O throughput, especially when low latency local storage is not available or desirable (cloud environments, scalable enterprise storage, etc.).

Optimized Columns allow the Intel Arria 10 GX FPGA with Swarm64 DA to access only the data relevant to the query, as shown in the illustration example below. On the left side of the picture the use of a multi-part key (ws_order_number, ws_sold_date) is illustrated. This includes searching through the index, and then fetching a number of scattered blocks of data from storage to finish the query processing. Each block contains many rows. This effect is often referred to as Read Amplification – when these blocks fetch many rows which are not relevant to the query and are retrieved together with the comparatively few relevant rows.

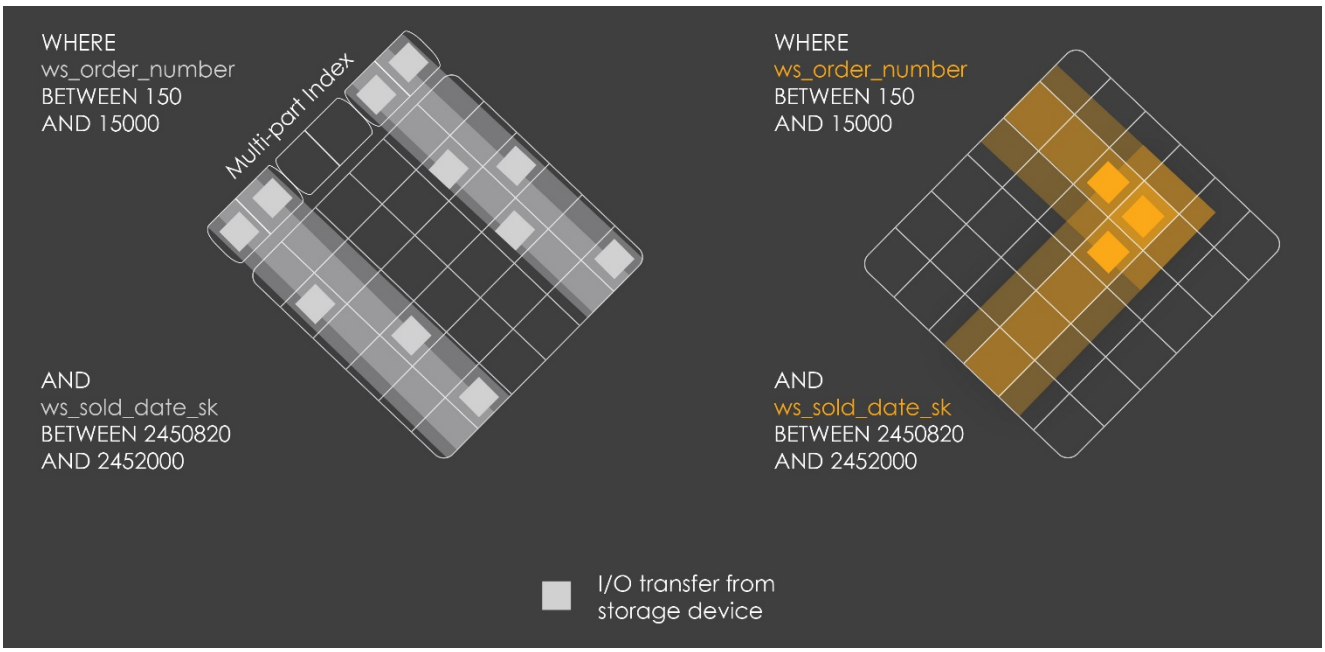

Figure 2. An example of an Optimized Column's data layout scheme

As shown above, the mechanism arranges the data approximately along the column ranges of each of the Optimized Columns. When querying for one or multiple ranges along the Optimized Columns, only the data at the intersection of each range is retrieved, leading to much fewer I/O transfers from storage device and much lower Read Amplification. The approximate data layout is then filtered with the help of the Intel Arria 10 GX FPGA and processed further in the database.

The approximate data layout is iteratively refined with a proprietary mechanism (pat. pend. via Swarm64). The parameters of this refinement operation, the Optimizer, can be configured as outlined below.

Contrasting to native indexes (such as multi-part keys), Optimized Columns require very little storage space. In a typical PostgreSQL database set-up for data warehousing the indexes may use between 50%-70% of the required storage space. Optimized Columns typically amount to 2% or less. For more details on Optimized Columns implementation, refer to Appendix B: Database Architecture Guidelines for the Intel Arria 10 GX FPGA + Swarm64 Data Accelerator.

Multi-OS support

With all these benefits and capabilities available on the Intel Arria 10 GX FPGA with Swarm64 DA on PostgreSQL, organizations can now choose their deployments' environments. The Intel Arria 10 GX FPGA with Swarm64 DA on PostgreSQL is now available on container platforms like Docker on Red Hat® Enterprise Linux®.

HPE ProLiant DL380 Gen10 server

The HPE ProLiant DL380 Gen10 server delivers the latest in security, performance, and expandability. It supports the Intel Xeon Processor Scalable Family with up to a 71% performance gain and 27% increase in cores,[1] and the HPE 2666 MT/s DDR4 SmartMemory supports 3.0 TB and up to 11%[2] faster than 2400 MT/s memory.

The HPE ProLiant DL380 Gen10 server has an adaptable chassis, including HPE modular drive bay configuration options with up to 30 SFF, up to 19 LFF, or up to 20 NVMe drive options along with support for up to three double-wide GPU options. Along with an embedded 4x1GbE, there is a choice of HPE FlexibleLOM or PCIe standup adapters, which offer a choice of networking bandwidth (1GbE to 40GbE) and fabric that adapt and grow to changing business needs.

The HPE ProLiant DL380 Gen10 server comes with a complete set of HPE Technology Services, delivering confidence, reducing risk, and helping customers realize agility and stability.

Figure 3. HPE ProLiant DL380 Gen10 server

Solution components

This paper provides configuration and performance information for Intel Arria 10 GX FPGA with Swarm64 DA in a PostgreSQL database environment. The tests were run on an HPE ProLiant DL380 Gen10 server running Red Hat 7.4 with Docker.

[1] Intel measurements. Up to 71% performance increase of Intel Xeon Platinum vs. previous generation E5 v4 average performance based on key industry-standard benchmark calculations submitted by OEMs comparing two-socket Intel Xeon Platinum 8180 to E5-2699 v4 family processors. Any difference in system hardware or software design or configuration may affect actual performance. May 2017. Up to 27% performance increase of Intel Xeon Platinum vs. previous generation comparing two-socket Intel Xeon Platinum 8180 (28 cores) to E5-2699 v4 (22 cores). Calculation 28 cores/22 cores= 1.27 = 27%. May 2017.

[2] The Gen10 2666 MT/s memory speed is 11% faster than that of Gen9 2400 MT/s, enabling faster server performance.

Hardware

Table 1 lists the details of the hardware configurations tested, as well as the platforms with traditional storage that were used for comparison testing. For more details, see Appendix A: Bill of materials.

Table 1. Hardware configurations used for testing solutions with the Intel Arria 10 GX FPGA + Swarm64 DA and with standard PostgreSQL

Qty	HPE ProLiant DL380 Gen10 with Intel Arria 10 GX FPGA + Swarm64DA	HPE ProLiant DL380 Gen10
Processor	2 x Intel Xeon Gold 6130 CPU @ 2.10GHz	2 x Intel Xeon Gold 6130 CPU @ 2.10GHz
Memory	384GB memory 12 x 32GB HPE DDR4 SmartMemory RDIMMs	384GB memory 12 x 32GB HPE DDR4 SmartMemory RDIMMs
Ethernet	HPE Ethernet 1Gb 4-port 331i Adapter - NIC	HPE Ethernet 1Gb 4-port 331i Adapter - NIC
OS drive	2 x HPE 960GB SATA MU SFF SSDs	2 x HPE 960GB SATA MU SFF SSDs
Data + Log drive	4 x HPE 3.84TB SATA RI SFF SC DS SSDs	4 x HPE 3.84TB SATA RI SFF SC DS SSDs

Software

- RHEL 7.4 (for the Intel Arria 10 GX FPGA + Swarm64 DA and PostgreSQL tests)
- PostgreSQL 11.x for Linux

Results

Performance results. The normalized Qph results for the ProLiant DL380 Gen10 with Intel Arria 10 GX FPGA + Swarm64 DA on PostgreSQL were compared to normalized Qph results from the ProLiant DL380 Gen10 with standard PostgreSQL. The ProLiant DL380 Gen10 with Intel Arria 10 GX FPGA + Swarm64 DA processed up to **2.7x faster** than the ProLiant DL380 Gen10 without Intel Arria 10 GX FPGA + Swarm64 DA at both 300GB and 1000GB scale factors. This demonstrates that customers moving from hardware without Intel Arria 10 GX FPGA + Swarm64 DA can expect significantly improved throughput when deploying with Intel Arria 10 GX FPGA + Swarm64 DA on Gen10 servers. For more details on test data, see Appendix D: Test and price data.

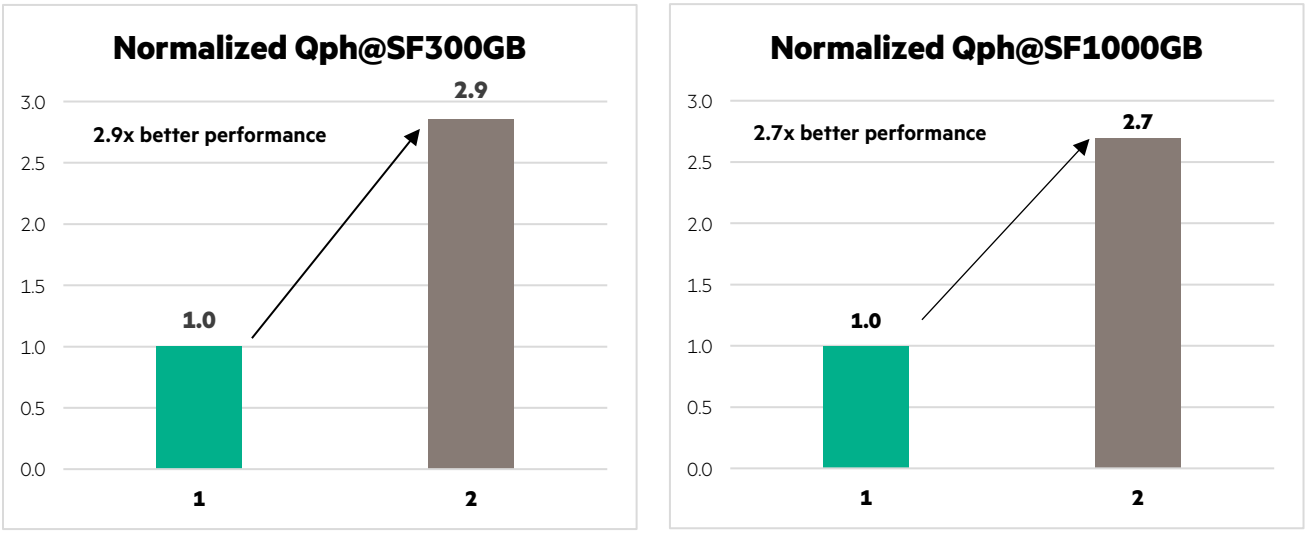

Figure 4. Normalized performance comparison of two scale factors with the ProLiant DL380 Gen10 with PostgreSQL + Intel Arria 10 GX FPGA + Swarm64 DA and with standard PostgreSQL

TCO-performance results. The ProLiant DL380 Gen10 with Intel Arria 10 GX FPGA with Swarm64 DA on PostgreSQL also offers a significant reduction in hardware costs as compared to the ProLiant DL380 Gen10 with standard PostgreSQL. Figure 5 below shows the normalized TCO-performance cost of each hardware configuration at both 300GB and 1000GB scale factors with the ProLiant DL380 Gen10 with Intel Arria 10 GX FPGA + Swarm64 DA on PostgreSQL coming in at up to **41% lower total cost for a given throughput**.

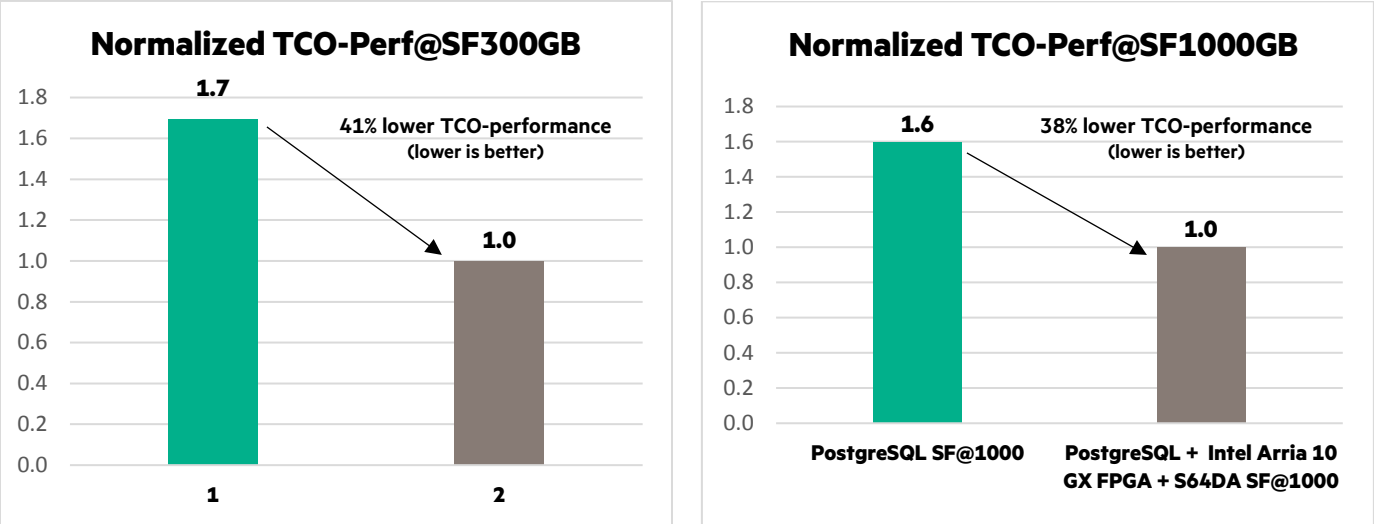

Figure 5. Normalized TCO-performance comparison of two scale factors with the ProLiant DL380 Gen10 with PostgreSQL+ Intel Arria 10 GX FPGA + Swarm64 DA and with standard PostgreSQL

Individual query testing

The individual query tests were run on an HPE ProLiant DL380 Gen10 server to demonstrate the benefits of when the data is not already cached. For these tests, the queries were run on an HPE ProLiant DL380 Gen10 server with Intel Arria 10 GX FPGA + Swarm64 DA and with standard PostgreSQL.

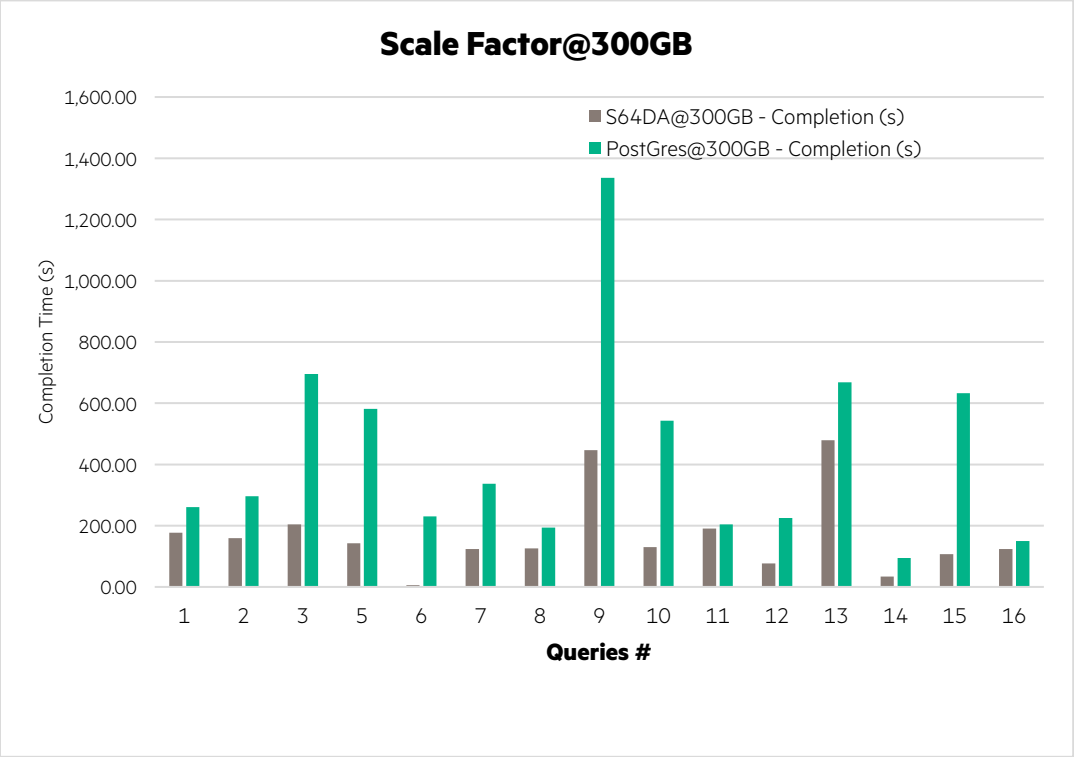

Figure 6. Number of queries for HPE ProLiant DL380 Gen10 with Intel Arria 10 GX FPGA + Swarm64 DA and standard PostgreSQL with SF@300GB

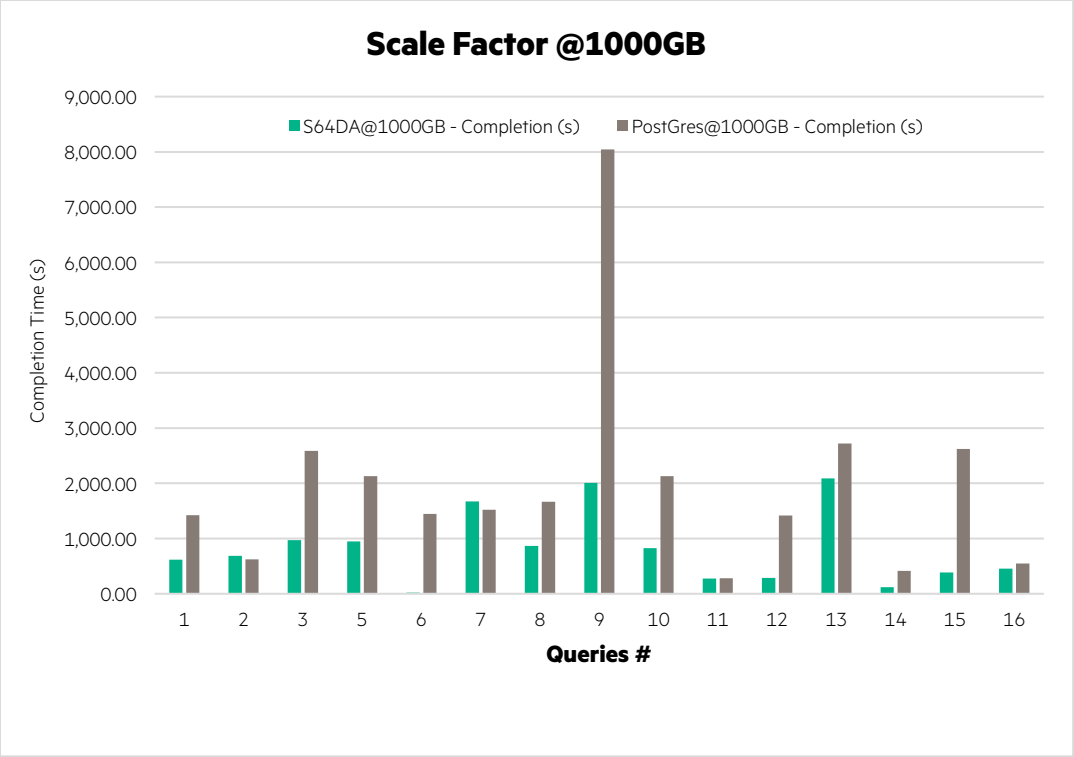

Figure 7. Number of queries for the HPE ProLiant DL380 Gen10 with Intel Arria 10 GX FPGA + Swarm64 DA and standard PostgreSQL with SF@1000GB

For more details on queries, see Appendix D: Test and price data.

Summary

The winning combination of Intel Arria 10 GX FPGA + Swarm64 DA with PostgreSQL on HPE ProLiant DL380 Gen10 provides world-leading results of up to 2.7x the performance with an excellent TCO-performance of up to 41% lower cost, for a given throughput, than with standard PostgreSQL. Customers who are moving from older hardware can expect substantial cost reduction and performance improvements, including much faster query completion for I/O intensive queries. Environments where Linux is required can now deploy PostgreSQL with Intel Arria 10 GX FPGA + Swarm64 DA, and have lower-cost alternatives compared to standard PostgreSQL without Intel Arria 10 GX FPGA + Swarm64 DA.

This Reference Configuration describes solution testing was performed in August and September 2018.

Implementing a proof-of-concept

As a matter of best practice for all deployments, HPE recommends implementing a proof-of-concept using a test environment that matches as closely as possible to the planned production environment. In this way, appropriate performance and scalability characterizations can be obtained. For help with a proof-of-concept, contact an HPE Services representative (hpe.com/us/en/services/consulting.html) or your HPE partner.

Appendix A: Bill of materials

The tables below show the system under test BOM for evaluating Intel Arria 10 GX FPGA + Swarm64 DA.

Note

Part numbers are at time of testing and subject to change. The bill of materials does not include complete support options or other rack and power requirements. If you have questions regarding ordering, please consult with your HPE Reseller or HPE Sales Representative for more details. hpe.com/us/en/services/consulting.html

Table 2. Cost Swarm64 Gen10 as of November 27, 2018

ProLiant DL Gen10 300GB and 1000GB Configurations	Price Key	Part Number	Unit Price	Qty	Extended Price	3 Yr SW Price	3 Yr Maint Price
Management Nodes							
HPE DL380 Gen10 24SFF CTO Server	1	868704-B21	$2,759	1	$2,759		
HPE DL380 Gen10 High Perf Fan Kit	1	867810-B21	$239	1	$239		
HPE DL380 Gen10 2SFF Bay Kit	1	826687-B21	$249	1	$249		
HPE DL380 Gen10 Intel Xeon-Gold 6130 FIO Processor Kit	1	826866-L21	$2,999	1	$2,999		
HPE DL380 Gen10 Intel Xeon-Gold 6130 Processor Kit	1	826866-B21	$2,999	1	$2,999		
HPE 32GB 2Rx4 PC4-2666T-R Kit	1	815100-B21	$1,230	12	$14,760		
HPE 960GB SATA MU SFF SC DS SSD	1	877782-B21	$1,659	2	$3,318		
HPE 3.84TB SATA RI SFF SC DS SSD	1	877764-B21	$5,199	4	$20,796		
Intel Arria 10GX FPGA	1	Q9B37A	$7,499	1	$7,499		
HPE 3Y FC 24x7 DL360 Gen10 SVC	1	H8QF0E	$1,811	1			$1,811
HPE iLO Adv incl 3yr TS U E-LTU	1	E6U64ABE	$469	1		$469	
HP W1972a 18.5-In LED Monitor (1 + 2 spare)	1	B7M13A8#ABA	$79	3	$237		
HP PS/2 Keyboard And Mouse Bundle (1 + 2 spare)	1	B1T13AA#ABA	$27	3	$81		
				Subtotal	**$55,936**	**$469**	**$1,811**
Network							
HPE 1620-24G Switch	1	JG913A	$299	1	$299		
				Subtotal	$299	$0	$0
Rack							
HPE 42U 600x1075mm Adv G2 Kit Pllt Rack	1	P9K07A	$1,179	1	$1,179		
HPE 24A High Voltage Core Only Corded PDU	1	252663-D74	$259	1	$259		
				Subtotal	**$1,438**	**$0**	**$0**
		Total HW Extended Price			**$57,673**	**$469**	**$1,811**
Hardware + Maintenance		**Total HW Discounts**			**$0**	**$0**	**$0**
		Total HW			**$57,673**	**$469**	**$1,811**
External Server Software							
Swarm64 DA License	2	Swarm64 Monthly License	$1,200	36		$43,200	
PostgreSQL 10	3	External	$0	1		$0	
RHEL Svr 2 Sckt/2 Gst 3yr 24x7 E-LTU	1	G3J30AAE	$3,702	1		$3,702	
				Subtotal	**$0**	**$46,902**	**$0**
Price Key: 1 - HPE	**Grand Total**				**$57,673**	**$47,371**	**$1,811**

All discounts are based on US list prices and for similar quantities and configurations. A 40% discount was based on the overall specific components pricing from vendor 1 in this single quotation. Discounts for similarly sized configurations will be similar to those quoted here, but may vary based on the components in the configuration.	**3 year cost of ownership USD:**	**$106,855**

	Swarm64DA Qph @ 300GB	**$9,066**
	Swarm64DA Qph @ 1000GB	**$6,817**
	Swarm64DA $ USD/Qph @ 300GB	**$0.33**
	Swarm64DA $ USD/Qph @ 1000GB	**$0.44**

Sales contact: HPE WW Headquarters, 3000 Hanover St., Palo Alto, CA 94304-1185 (650) 857-1501 or HPE: 855-472-5233

Table 3. Cost PostgreSQL Gen10 as of November 27, 2018

ProLiant DL Gen10 300GB and 1000GB Configurations	**Price Key**	**Part Number**	**Unit Price**	**Qty**	**Extended Price**	**3 Yr SW Price**	**3 Yr Maint Price**
Management Nodes							
HPE DL380 Gen10 24SFF CTO Server	1	868704-B21	$2,759	1	$2,759		
HPE DL380 Gen10 High Perf Fan Kit	1	867810-B21	$239	1	$239		
HPE DL380 Gen10 2SFF Bay Kit	1	826687-B21	$249	1	$249		
HPE DL380 Gen10 Intel Xeon-Gold 6130 FIO Processor Kit	1	826866-L21	$2,999	1	$2,999		
HPE DL380 Gen10 Intel Xeon-Gold 6130 Processor Kit	1	826866-B21	$2,999	1	$2,999		
HPE 32GB 2Rx4 PC4-2666T-R Kit	1	815100-B21	$1,230	12	$14,760		
HPE 960GB SATA MU SFF SC DS SSD	1	877782-B21	$1,659	2	$3,318		
HPE 3.84TB SATA RI SFF SC DS SSD	1	877764-B21	$5,199	4	$20,796		
Intel Arria 10GX FPGA	1	Q9B37A	$7,499	1	$7,499		
HPE 3Y FC 24x7 DL360 Gen10 SVC	1	H8QF0E	$1,811	1			$1,811
HPE iLO Adv incl 3yr TS U E-LTU	1	E6U64ABE	$469	1		$469	
HP W1972a 18.5-In LED Monitor (1 + 2 spare)	1	B7M13A8#ABA	$79	3	$237		
HP PS/2 Keyboard And Mouse Bundle (1 + 2 spare)	1	B1T13AA#ABA	$27	3	$81		
				Subtotal	**$55,936**	**$469**	**$1,811**
Network							
HPE 1620-24G Switch	1	JG913A	$299	1	$299		
				Subtotal	**$299**	**$0**	**$0**
Rack							
HPE 42U 600x1075mm Adv G2 Kit Pllt Rack	1	P9K07A	$1,179	1	$1,179		
HPE 24A High Voltage Core Only Corded PDU	1	252663-D74	$259	1	$259		
				Subtotal	**$1,438**	**$0**	**$0**
Hardware + Maintenance		**Total HW Extended Price**			**$57,673**	**$469**	**$1,811**
		Total HW Discounts			**$0**	**$0**	**$0**
		Total HW			**$57,673**	**$469**	**$1,811**

External Server Software							
PostgreSQL 10	3	External	$0	1		$0	
RHEL Svr 2 Sckt/2 Gst 3yr 24x7 E-LTU	1	G3J30AAE	$3,702	1		$3,702	**$0**
				Subtotal	**$0**	**$3,702**	**$0**
Price Key: 1 - HPE 2-SWARM64 AS 3-3rd Party		**Grand Total**			**$57,673**	**$4,171**	**$1,811**
		3 year cost of ownership USD:					**$63,655**
		PostgreSQL Qph @ 300GB					**3,176**
		PostgreSQL Qph @ 1000GB					**2,533**
		PostgreSQL $ USD/Qph @ 300GB					**$0.56**
		PostgreSQL $ USD/Qph @ 1000GB					**$0.70**

Sales contact: HPE WW Headquarters, 3000 Hanover St., Palo Alto, CA 94304-1185 (650) 857-1501 or HPE: 855-472-5233

Software

Table 4. Software and versions required to evaluate the Intel Arria 10 GX FPGA with Swarm64 Data Accelerator

Software	Version
OS	CentOS/RHEL 7.4
Docker	18.03.0-ce or later
Python	3.6
PostgreSQL	10.x or 11.x

Appendix B: Database architecture guidelines for the Intel Arria 10 GX FPGA + Swarm64 Data Accelerator

The Intel Arria 10 GX FPGA + Swarm64 DA: Engineered analytics

Intel Arria 10 GX FPGA with Swarm64 DA is an engineered solution to accelerate analytical and hybrid analytical transactional processing in PostgreSQL and other popular databases, such as MariaDB and MySQL, with the help of the Intel Programmable Accelerator Card. The solution consists of the following components, which are described in more detail below and in the following sections.

Table 5. Components of the Intel Arria 10 GX FPGA with Swarm64 DA

Function	Description
PostgreSQL version 10 and 11	The accelerated database. The Data Accelerator retains the existing database functionality and connectivity to PostgreSQL drivers (JDBC, ODBC …) and PostgreSQL compatible tools (e.g., Tableau).
Intel Programmable Accelerator Card (PAC)	A PCIe Gen3 x8 add-on card for the server. Intel Arria 10 GX FPGA with Swarm64 DA's solution is loaded as a configuration on the "Field Programmable Gate Array" (FPGA) of the PAC card. For further reference, refer to: intel.com/content/www/us/en/programmable/products/boards_and_kits/dev-kits/altera/acceleration-card-arria-10-gx.html
Intel Arria 10 GX FPGA + Swarm64 Data Accelerator for PostgreSQL	The accelerator solution, consisting of a software extension interacting with the configuration on the Intel PAC.
FPGA Configuration on the PAC	A specific, bandwidth and efficiency optimized configuration that co-processes database function and works cooperatively with the server CPUs to deliver the overall system performance.

Choosing Optimized Columns in a table

Optimized Columns should be used where querying for frequent data points or for data ranges is common. As a rule of thumb, consider Optimized Columns for:

- Time or date ranges
- Price ranges or a specific price point
- Quantities
- Key business metrics in a fact table tracking or sensor data
- Spatial information (e.g., x/y, longitude/latitude)

Unlike multi-part keys (also called compound keys or multi-part indexes) in the native database, Optimized Columns can be queried independently from each other and in any order (the order in which they are defined has no impact). The more Optimized Columns are queried, the faster the query executes. Please note that this behavior is very different from the native multi-part keys, which perform best if they are used in order and skipping earlier key parts carries a heavy performance penalty. Optimized Columns benefit from the aforementioned approximate data layout along the Optimized Column ranges.

Between one and three Optimized Columns can be set per table. One is always required. Intel Arria 10 GX FPGA with Swarm64 DA recommends to use two or three optimized columns per table. There is a small performance benefit using two instead of three, but only in cases where queries utilizing the third column are very rare.

As a general rule, Optimized Columns should not be used for columns that are not queried by range, such as IDs typically used in primary and foreign keys. Worst case, this may use one of the three Optimized Column slots otherwise usable for a more suited column.

Primary, Unique, and Foreign Key Hints

Primary, Unique, and Foreign Key Hints have the same purpose as in native PostgreSQL tables, with the difference that they're only hints and thereby not enforced. Primary Key Hints also encompass uniqueness. It's important to keep in mind that erroneously adding a Unique or Primary Key Hint to a column that contains duplicates can lead to wrong query results. See the following example.

```
CREATE FOREIGN TABLE t0(col0 INT NOT NULL, col1 TEXT) SERVER
swarm64da_server OPTIONS(optimized_columns 'col0');
CREATE FOREIGN TABLE t1(col0 INT NOT NULL, col1 TEXT) SERVER
swarm64da_server OPTIONS(optimized_columns 'col0');
INSERT INTO t0 VALUES (0, 'A'), (1, 'B'), (2, 'C'), (3, 'D'), (4, 'E');
INSERT INTO t1 VALUES (0, 'a'), (1, 'b'), (1, 'c'), (1, 'd'), (4, 'e');
ANALYZE t0;
ANALYZE t1;

SELECT * FROM t0 INNER JOIN t1 ON t0.col0 = t1.col0;
ALTER FOREIGN TABLE t1 OPTIONS(ADD unique_keys 'UNIQUE(col0)'); SELECT * FROM
t0 INNER JOIN t1 ON t0.col0 = t1.col0;
```

Table `t1` contains duplicates. Joining `t1` with `t0` yields wrong results as soon as a Unique Key Hint is added to `t1.col0`.

```
user=# SELECT * FROM t0 INNER JOIN t1 ON t0.col0 = t1.col0;

     col0 | col1 | col0 | col1
     ------+------+------+------

          0 | A    |   0  | a
          1 | B    |   1  | b
          1 | B    |   1  | c
          1 | B    |   1  | d
          4 | E    |   4  | e


 (5 rows)

user=# ALTER FOREIGN TABLE t1 OPTIONS(ADD unique_keys 'UNIQUE
(col0)');
ALTER FOREIGN TABLE

 user=# SELECT * FROM t0 INNER JOIN t1 ON t0.col0 = t1.col0;
       col0 | col1 | col0 | col1
      ------+------+------+------
          0 | A    |    0 | a
          1 | B    |    1 | b
          4 | E    |    4 | e

(3 rows)
```

For more information on how to create and alter Primary, Unique, and Foreign Key Hints, see the respective section below.

Choosing between native tables and Intel Arria 10 GX FPGA accelerated tables

Native tables and foreign tables can be freely combined. Below is an illustration of the mix depending on the use case.

Some rules of thumb for choosing between native tables and Intel Arria 10 GX FPGA with Swarm64 DA tables: Choose native tables for the database tables in the following cases:

- You interact with the table in many small transactions of a few or a few hundred rows. Deletes of or Updates to single values or a small fraction of the table at a time is required. You require SERIALIZABLE as a Transaction Scheme
- Your workload often includes retrieving single values from an indexed column. Choose Accelerated tables in the following cases:
  - Your tables are very large or intended to become very large over time
  - You are typically querying ranges (e.g., time ranges)
  - You have Fact tables in your data warehouse
  - When data is moving a high velocity (e.g., streaming into the database)
  - When logging events
  - For large, mostly static, tables

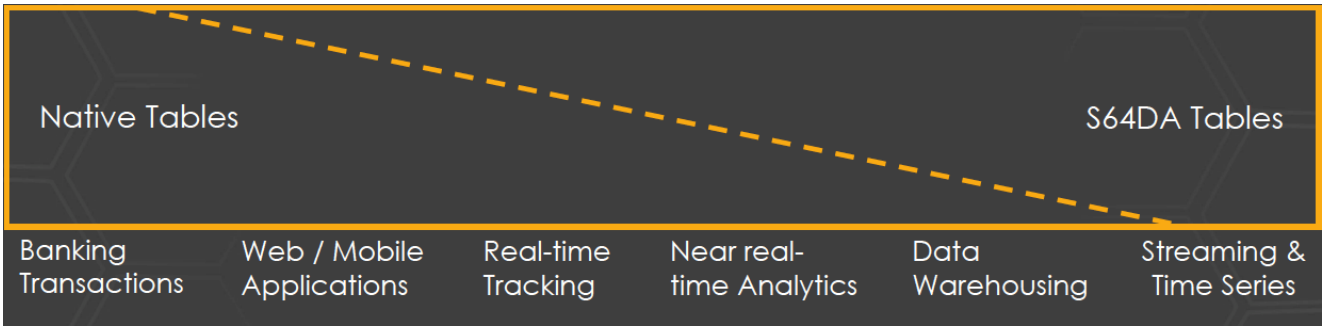

Figure 8. Native tables as compared to Intel FPGA with Swarm64 DA tables

Appendix C: Database configuration

Server configuration sysctl settings

```
kernel.numa_balancing=0 // highly recommended for kernel 3.19.0-56

vm.dirty_background_bytes = 134217728

vm.swappiness=5. vm.max_map_count=65536 vm.zone_reclaim_mode=0
```

Memory and parallelism (performance)

A trade-off has to be made between parallelism (i.e., the number of workers allowed to work on a task) and the amount of `work_mem` given to each of them. `work_mem` is allocated per parallel worker and per query execution node, and every connection can spawn `max_parallel_workers_per_gather` until the `max_worker_processes` limit is reached. Empirically, providing around 1/16th of the total memory as `work_mem` per query node works well for 4 to 8 parallel workers, if the number of simultaneously executed heavy-weight queries is low (e.g., 2-5 at a time). For heavy analytics workloads on large datasets it may be desirable to increase `max_parallel_workers_per_gather` to 16, in which case 1/16th to 1/32nd of the available RAM shall be used as `work_mem` and limit the `max_worker_processes` – the overall maximum – to less than 32. This will maximize resource usage when a single query runs at a time.

```
work_mem = 16GB //choose 1/16th of total system RAM, up to 1/8th in certain cases

effective_cache_size = 128GB //this parameter is an estimate for planning only. Set between 1/2 and 1/4th of total system RAM

shared_buffers = 64GB //choose between 1/4th and 1/8th of total system RAM

maintenance_work_mem =6GB

max_connections = 32

max_worker_processes = 30

force_parallel_mode = False max_parallel_workers_per_gather=16

parallel_setup_cost=500

dynamic_shared_memory_type = posix max_stack_depth = 7MB
```

Intel Arria 10 GX FPGA with Swarm64 DA specific

```
shared_preload_libraries = 'Swarm64 DA.so'

swarmdb.fpga_backend = SDA

swarmdb.enable_optimization = true

swarmdb.max_optimization_mb_per_second = 512
```

Optimizer configuration

The optimizer has an I/O budget which it can spend to optimize tables. There's a global budget that can be set in `postgres.conf` as `swarm64da.max_optimization_mb_per_second = 64`. The budget can be overwritten per database via:

```
ALTER DATABASE db SET Swarm64 DA.max_optimization_mb_per_second TO 128;
```

There's an optimizer background worker running for every DB. Already running optimization workers must be explicitly informed about changes to the configuration. This is usually done by sending a `SIGHUP` signal to the respective process, or to `postmaster` which broadcasts the signal to all worker processes. Usually this is done via `pg_ctl_reload`.

The optimizer can be inspected by running `SELECT * from Swarm64DA.get_optimization_infos()`. It will return a table with one row per Intel Arria 10 GX FPGA with Swarm64 DA table in the current DB with various statistics on the optimization progress. Especially important are `optimization_level` and `optimization_target`.

The `optimization_target` describes how optimal a table has to be before the optimizer stops optimizing it. The `optimization_target` can either be specified during table creation or altered later. Its valid range is from 0 to 1000. 1000 is the absolute optimum. The `optimization_level` describes how well the table is currently optimized.

Data Definition (DDL)

Creating a table

Optimized Columns

The set of Optimized Columns is the only required option when creating an Intel Arria 10 GX FPGA with Swarm64 DA table. The Optimized Columns cannot be changed anymore after the table is created. At maximum three Optimized Columns are recommended. The following list shows which column types can be optimized. Especially variable length column types and fixed length column types greater than 8 bytes cannot be optimized at present. Optimized columns can be nullable.

Table 6. Column types that can be optimized

Column Type	Notes
BOOLEAN	Supported but not recommended because of low selectivity
SMALLINT INT/ INTEGER BIGINT SMALLSERIAL SERIAL	
DATE	The accelerator solution, consisting of a software extension interacting with the configuration on the Intel PAC
TIME	A specific, bandwidth and efficiency optimized configuration that co-processes database function and works cooperatively with the server CPUs to deliver the overall system performance.
TIMESTAMP/ TIMESTAMPTZ	
ENUM	Only equality comparison supported
NUMERIC/ DECIMAL	At maximum 13 digits

Custom types are supported as Optimized Columns as long as they have a fixed length less than or equal to eight bytes and are comparable using a standard integer comparison operation.

Creating a table with two Optimized Columns is done as follows:

```
CREATE FOREIGN TABLE my_table (col0 INT, col1 TIMESTAMP NOT NULL,
col2 VARCHAR(30)) SERVER Swarm64 DA_server OPTIONS (optimized_columns 'col0, col1');
```

Optimization level target

The optimization level target specifies how much the table will be optimized. The value is between 0 and 1000, where 1000 means fully optimized and 0 means not optimized at all. The optimization level target is used to control how much of the optimizer's I/O budget is spent on a table. It doesn't help to optimize tables which are, e.g., accessed very infrequently or always only in large chunks. For more information, see section Optimizer configuration. The optimization target is specified like:

```
CREATE FOREIGN TABLE my_table (col0 INT, col1 TIMESTAMP NOT NULL,
col2 VARCHAR(30)) SERVER Swarm64 DA_server OPTIONS (optimized_columns 'col0, col1',
optimization_level_target '800');
```

Primary, Unique, and Foreign Key inheritance

When used in conjunction with table inheritance, unique and primary keys are not inherited. This is the case for both inheriting from PostgreSQL native tables and from Intel Arria 10 GX FPGA with Swarm64 DA tables. These have to be added manually. Limitations of PostgreSQL on the number of columns being supported to be part of a composite index apply (`INDEX_MAX_KEYS`, by default 32). The syntax matches the syntax of PostgreSQL's Primary, Unique, and Foreign Key constraints.

Creating a Unique Key Hint

Per table, an arbitrary number of unique key hints may be defined.

Creating a Foreign Key Hint

Knowledge about join dependencies is important for query planning. The reason for this is that it signals to the query planner that for each row in the base table there is at least one row in the joined table. Using this information allows the planner to create a much more accurate estimate of the number of rows that are produced by the join and thus helps to find a query plan that is better suited for the data being processed.

Foreign Key Hints are non-enforced foreign key relationships between tables and are evaluated during query planning to improve the performance of joins involving an Intel Arria 10 GX FPGA with Swarm64 DA table as a base table. Referenced may be any table, either a PostgreSQL native or an Intel Arria 10 GX FPGA with Swarm64 DA table. In the columns list, an arbitrary number of columns may be specified using a comma as a delimiter. Foreign Key Hints prevent users from dropping a referenced native PostgreSQL or Swarm64 DA table if `CASCADE` keyword is not given in the `DROP (FOREIGN) TABLE` command.

```
CREATE FOREIGN TABLE base1(id INT, payload INT) SERVER
swarm64da_server OPTIONS(optimized_columns 'id'); CREATE FOREIGN
TABLE base2(id INT, payload INT) SERVER
swarm64da_server OPTIONS(optimized_columns 'id');
CREATE FOREIGN TABLE test_table(base1_id INT, base2_id INT, payload INT)
   SERVER swarm64da_server OPTIONS(optimized_columns
   'base1_id, base2_id',
   foreign_keys 'FOREIGN KEY (base1_id) REFERENCES base1(id), FOREIGN KEY (base2_id)
REFERENCES base2(id)');
```

Restrictions

- Contrary to PostgreSQL foreign key constraint specification, column lists may not be omitted and are not inferred by Intel Arria 10 GX FPGA with Swarm64 DA.
- Columns referenced in a foreign key hint must be of the same datatype as the referencing column.

Due to the nature of the non-enforced relationship, keywords such as CASCADE or RESTRICT are not supported.

Inspecting Hints

Unique, Primary, and Foreign Key Hints can be inspected by describing a table in the PostgreSQL client.

```
user=# \d t00
                          Foreign table "public.t00"
       Column |     Type     | Collation | Nullable | Default | FDW options
        --------+---------+-----------+----------+---------+-------------



FDW options: (optimized_columns 'c0', optimization_level_target
'1000', unique_keys 'UNIQUE (c1)')


user=# \d t01
                          Foreign table "public.t01"
       Column |     Type     | Collation | Nullable | Default | FDW options
        --------+---------+-----------+----------+---------+-------------



FDW options: (optimized_columns 'c0', optimization_level_target
'1000', primary_key 'PRIMARY KEY (c1)')
```

Creating table, Optimized Columns and Hints at the same time

All previously shown table options can be specified at the same time when creating a table. The following listing shows an example of creating a complex table.

```
CREATE FOREIGN TABLE orders (
   o_orderkey bigint NOT NULL, o_custkey int NOT
   NULL, o_orderstatus character(1) NOT NULL,
   o_totalprice numeric(13,2) NOT NULL,
   o_orderdate date NOT NULL,
   o_orderpriority character(15) NOT NULL, o_clerk
   character(15) NOT NULL, o_shippriority int NOT
   NULL,
   o_comment character varying(79) NOT NULL
)
SERVER swarm64da_server
OPTIONS(
   optimized_columns 'o_orderdate',
   optimization_level_target '900', primary_key
   'PRIMARY KEY (o_orderkey)',
   foreign_keys 'FOREIGN KEY (o_custkey) REFERENCES customer (c_custkey)',
   unique_keys 'UNIQUE(o_custkey)'
);
```

Analyzing tables to update statistics

The query optimizer relies on up-to-date statistics to find the best possible query plan. Table statistics can be updated with the `ANALYZE my table;` command. Just running `ANALYZE;` updates the statistics of all native PostgreSQL tables, but not of any foreign tables. To update statistics of foreign tables, you must explicitly list the tables in the `ANALYZE` command.

Optimized Columns

Currently Optimized Columns can only be specified when creating a table and not be altered afterwards. If you must change your Optimized Columns, we recommend creating a new table by copying the data over via:

```
INSERT INTO new_table SELECT * FROM old_table;
```

Hints

Hints can be added if there weren't any defined previously.

Hints may be set and reset when the hint was already defined. Keep in mind that you are required to list all hint parts when using `SET`, omitted hints are removed and not present anymore after executing the command.

```
ALTER FOREIGN TABLE table OPTIONS(SET unique_keys 'UNIQUE(c1)'); ALTER FOREIGN TABLE
table OPTIONS(SET primary_key 'PRIMARY KEY(c1)');
ALTER FOREIGN TABLE table OPTIONS(SET foreign_keys 'FOREIGN KEY (base1_id)
REFERENCES base1(id)');
```

Hints can be dropped if they were defined before.

```
ALTER FOREIGN TABLE table OPTIONS(DROP unique_keys); ALTER FOREIGN
TABLE table OPTIONS(DROP primary_key); ALTER FOREIGN TABLE table
OPTIONS(DROP foreign_keys);
```

For additional information on how to work with foreign tables in PostgreSQL, refer to the PostgreSQL documentation on CREATE FOREIGN TABLE, ALTER FOREIGN TABLE, DROP FOREIGN TABLE and DDL FOR FOREIGN DATA.

Data Query (DQL)

Swarm64 supports the full set of PostgreSQL functionality in native tables and a large sub-set in Swarm64 DA tables. For a full list of possible SQL queries, refer to postgresql.org/docs/current/static/sql.html.

In general, applications querying the data require no changes to their SQL query strings. However, there are a few restrictions to keep in mind. When such a restriction is affecting the queries in question, it is recommended to use the database's native table format. As Swarm64 can freely join between different table formats, this is the preferred solution.

Swarm64 DA tables are intended for (near-real-time) data warehousing, time-series data, data vaults, streaming, data logging, or as a recipient of Extract-Transform-Load (ETL) results. It is append-only, which means the current version does not provide the ability to change or delete individual records. This does not limit the ability to batch load, real-time log or execute Extract-Transform-Load or Extract-Load-Transform processes. For transactional workload (OLTP) it is strongly recommended to use the native database tables. For Hybrid Transactional Analytical Processing, it is recommended to combine native tables for transactions with Swarm64 DA tables for analytics.

```
ALTER FOREIGN TABLE table OPTIONS(ADD unique_keys 'UNIQUE(c1)'); ALTER FOREIGN TABLE
table OPTIONS(ADD primary_key 'PRIMARY KEY(c1)');
ALTER FOREIGN TABLE table OPTIONS(ADD foreign_keys 'FOREIGN KEY (base1_id)
REFERENCES base1(id)');
```

Appendix D: Test and price data

Cost

A key factor in reducing TCO is the inclusion of features with Intel Arria 10 GX FPGA with Swarm64 DA on PostgreSQL which comparable database solutions either provide at additional cost or simply do not offer.

	Completion time	SF (GB)	Performance Qph@SF
PostgreSQL SF @300	6,449.3	300	3,175.5
PostgreSQL + Swarm64DA SF @300	2,524.6	300	9,065.8
PostgreSQL SF @1000	31,124.6	1,000	2,532.7
PostgreSQL + Swarm64DA SF @1000	12,218.3	1,000	6,817.5

	SF (GB)	Performance Qph@SF	Normalized Qph@SF	TCO-Perf	Normalized TCO-Perf	% TCO-Perf
PostgreSQL SF@300	300	3,175.5	1.0	$0.55	1.7	169%
PostgreSQL + Swarm64DA SF@300	300	9,065.8	2.9	$0.33	1.0	
PostgreSQL SF@1000	1,000	2,532.7	1.0	$0.69	1.6	160%
PostgreSQL + Swarm64DA SF@1000	1,000	6,817.5	2.7	$0.43	1.0	

Best practices and configuration guidance for the solution

HPE ProLiant DL380 Gen10 server

Initial setup for the HPE ProLiant DL380 Gen10 server consisted of various BIOS and SQL settings. The following BIOS settings were modified from the default settings:

- WorkloadProfile=Custom
- CollabPowerControl=Disabled
- EnergyPerfBias=MaxPerf
- IntelDmiLinkFreq=Auto
- IntelNicDmaChannels=Enabled
- IntelPerfMonitoring=Disabled
- IntelProcVtd=Disabled
- IntelUpiFreq=Auto
- IntelUpiLinkEn=Auto
- IntelUpiPowerManagement=Disabled
- IntelligentProvisioning=Enabled

- LlcPrefetch=Enabled
- ThermalConfig=MaxCooling

Capacity and sizing

Workload description

Two separate types of tests were conducted using a subset of Business Intelligence/Data Analytics with a TPC-H-like set of 15 queries on PostgreSQL database, and the second with the same queries on PostgreSQL + Intel Arria 10 GX FPGA with Swarm64 DA configuration.

Business Intelligence/Data Analytics

The Business Intelligence/Data Analytics workload was utilized for this effort. It consists of a suite of business oriented ad-hoc queries and concurrent data modifications. The queries and the data populating the database have been chosen to have broad industry-wide relevance. This benchmark illustrates decision support systems that examine large volumes of data, execute queries with a high degree of complexity, and give answers to critical business questions. The performance metric reported is the sum of the completion time of all queries running individually, and reflects performance acceleration of different categories of these queries. The tests include two different scale factors, 300GB and 1000GB. Essentially, we measure the query processing power with queries processing a single stream.

Individual query testing

For these tests, in order to eliminate the impact of caching, the caches were dropped prior to each query. Each individual query was run from script in Linux command line.

Analysis and recommendations

Metric and result

The single stream metric is the product of one hour in seconds (3600), the SF in GB, and the inverse of the geo-mean of the 15 queries involved as shown below:

$$Qph = \frac{3600 * SF\ (in\ GB)}{\sqrt[15]{\prod_{i=1}^{i=15} QI(i,0)}}$$

Where SF is the scale factor in GB, QI(i,0) is the timing interval, in seconds, of query Qi within the single query stream of the queries running individually, and Qph is the resulting metric used in this paper.

Resources and additional links

HPE Data and Analytics Reference Configurations, hpe.com/info/dm-ra

HPE ProLiant DL380 Gen10 server, hpe.com/servers/dl380

Intel Arria 10 GX FPGA, intel.com/content/www/us/en/products/programmable/fpga/arria-10.html

Intel FPGA Acceleration Hub, intel.com/content/www/us/en/programmable/solutions/acceleration-hub/overview.html

To help us improve our documents, please provide feedback at hpe.com/contact/feedback.

a00061379enw, November 2018