processing task so that it can produce the task from the factory object later when it tries to perform the task next time. In some other embodiments, the data analytics application 150 places the data processing task into a queue of data processing tasks. The queue includes tasks that are not ready to be performed, e.g., due to lack of sufficient memory (and/or other types of needed computer resources), but can be performed when the memory becomes available.. Given that memory blocks locked for other tasks can be released once those tasks are completed, the memory 140 can have enough free space for the task later. The data analytics application 150 will check later whether the memory becomes available for the data processing task and can retrieve the data processing task by using the factory object or from the queue and perform the data processing task when the memory becomes available.

[0019] By using the factory object or the queue, the data analytics application 150 does not need to allocate any memory blocks for inputting or reading of the data of the task before the task can be performed. This way, no memory block is locked until there is enough available memory to perform the task. The data analytics application 150 can use the memory block, the locking of which is avoided, to perform other data processing tasks. For instance, in a situation where the memory has only 400 MB free space but 600 MB is needed to perform the task, conventional data analytic platforms typically lock a memory block of 400 MB for the task anyway and wait till another 200 MB becomes available to perform the task. The 400 MB is "wasted" during the time of waiting. Different from the conventional data analytic platforms, the data analytics application 150 does not lock the 400 MB. Rather, it uses the 400 MB to perform other data processing tasks that requires no more than 400 MB while it is waiting for the memory to have enough free space for the task. Thus, the memory management technique implemented by the data analytics application 150 optimizes usage of the memory 140 and improves data processing efficiency.

[0020] The memory management technique implemented by the data analytics application 150 also avoids the situation where multiple data processing tasks lock memory concurrently and therefore, prevents excessive usage of the memory 140. Also, as the factory object may have a memory footprint that is smaller than a memory footprint of the task, the data analytics application 150 further improves memory usage by storing the factory object, as opposed to the task itself, in the memory 140. Thus, the data analytics application 150 provides a more advantageous memory management technique than the conventional data analytic platforms.

[0021] In some embodiments, the memory 140 includes a primary memory providing relatively fast access to a relatively small data storage area and a secondary memory

[0063] FIG. 6 is a high-level block diagram illustrating a functional view of a typical computer system 600 for use as the data analytics system 110 of FIG. 1 according to an embodiment.

[0064] The illustrated computer system includes at least one processor 602 coupled to a chipset 604. The processor 602 can include multiple processor cores on the same die. The chipset 604 includes a memory controller hub 620 and an input/output (I/O) controller hub 622. A memory 606 and a graphics adapter 612 are coupled to the memory controller hub 620 and a display 618 is coupled to the graphics adapter 612. A storage device 608, keyboard 610, pointing device 614, and network adapter 616 may be coupled to the I/O controller hub 622. In some other embodiments, the computer system 600 may have additional, fewer, or different components and the components may be coupled differently. For example, embodiments of the computer system 600 may lack displays and/or keyboards. In addition, the computer system 600 may be instantiated as a rack-mounted blade server or as a cloud server instance in some embodiments.

[0065] The memory 606 holds instructions and data used by the processor 602. In some embodiments, the memory 606 is a random-access memory. The storage device 608 is a non-transitory computer-readable storage medium. The storage device 608 can be a HDD, SSD, or other types of non-transitory computer-readable storage medium. Data processed and analyzed by the data analytics system 110 can be stored in the memory 606 and/or the storage device 608.

[0066] The pointing device 614 may be a mouse, track ball, or other type of pointing device, and is used in combination with the keyboard 610 to input data into the computer system 600. The graphics adapter 612 displays images and other information on the display 618. In some embodiments, the display 618 includes a touch screen capability for receiving user input and selections. The network adapter 616 couples the computer system 600 to the network 160.

[0067] The computer system 600 is adapted to execute computer modules for providing the functionality described herein. As used herein, the term “module” refers to computer program instruction and other logic for providing a specified functionality. A module can be implemented in hardware, firmware, and/or software. A module can include one or more processes, and/or be provided by only part of a process. A module is typically stored on the storage device 608, loaded into the memory 606, and executed by the processor 602.

[0068] The particular naming of the components, capitalization of terms, the attributes, data structures, or any other programming or structural aspect is not mandatory or significant,

We claim:

1. A computer-implemented method, comprising:

accessing, by a computer system, an analytics workflow comprising a sequence of tools, each tool in the sequence of tools adapted to perform a data analytics function;

determining, at a tool in the analytics workflow at a first time, whether a memory associated with the computer system is available for storing data records for performing a data processing task;

in response to determining, at the first time, that the memory is not available for storing the data records for performing the data processing task, generating a factory object for the data processing task, the factory object specifying instructions to produce the data processing task;

registering, by the tool, the factory object with a scheduler;

in response to detection of a free thread by the scheduler, requesting a task from the factory object;

in response to the request, producing the data processing task by using the factory object;

determining, at a second time that is after the first time, that memory is available for performing a reduced scope of the data processing task and scheduling the reduced scope of the data processing task with the scheduler, the determining comprising:

determining that the memory for performing the data processing task is more than an amount of available memory;

determining a smaller amount of memory required for performing the reduced scope of the data processing task;

determining that the memory required for the reduced scope of the data processing task is smaller than the amount of available memory; and

scheduling the reduced scope of the data processing task with the scheduler; and

in response to the scheduling, storing the data records using the available memory and performing, by the computer system, the reduced scope of the data processing task to process the data records.

2. The computer-implemented method of claim 1, wherein the factory object comprises program instructions that when executed, cause a computer processor to produce the data processing task.

3. The computer-implemented method of claim 1, wherein the factory object has a memory footprint smaller than a memory footprint of the data processing task.

4. The computer-implemented method of claim 1, further comprising:

in response to determining, at the first time, that the memory is not available for storing the data records for performing the data processing task, placing the data processing task into a queue of data processing tasks; and

in response to determining, at the second time, that the memory is available for performing the reduced scope of the data processing task, removing the data processing task from the queue of data processing tasks.

5. The computer-implemented method of claim 4, further comprising:

determining, at a third time that is after the first time but before the second time, whether the memory is available for performing the data processing task; and

in response to determining, at the third time, that the memory is not available for performing the data processing task, placing the data processing task back into the queue.

6. The computer-implemented method of claim 1, wherein determining, at the first time, that the memory is not available for storing the data records for performing the data processing task comprises:

determining an amount of memory required for performing the data processing task;

determining an amount of memory that has not been reserved for other data processing tasks;

determining whether the amount of memory required for performing the data processing task is more than the amount of memory that has not been reserved for other data processing tasks; and

in response to determining that the amount of memory required for performing the data processing task is more than the amount of memory that has not been

reserved for other data processing tasks, determining that the memory is not available for performing the data processing task.

7. A computer system for managing memory in a computer, the computer system comprising:

a computer processor; and

a non-transitory computer-readable storage medium storing executable computer program instructions, the executable computer program instructions comprising instructions that when executed cause the computer processor to perform operations comprising:

accessing an analytics workflow comprising a sequence of tools, each tool in the sequence of tools adapted to perform a data analytics function;

determining, at a tool in the analytics workflow at a first time, whether a memory associated with the computer system is available for storing data records for performing a data processing task;

in response to determining, at the first time, that the memory is not available for storing the data records for performing the data processing task, generating a factory object for the data processing task, the factory object specifying instructions to produce the data processing task;

registering, by the tool, the factory object with a scheduler;

in response to detection of a free thread by the scheduler, requesting a task from the factory object;

in response to the request, producing the data processing task by using the factory object;

determining, at a second time that is after the first time, that memory is available for performing a reduced scope of the data processing task and scheduling the reduced scope of the data processing task with the scheduler, the determining comprising:

determining that the memory for storing the data records of the data processing task is more than an amount of available memory;

determining a smaller amount of memory required for performing the reduced scope of the data processing task;

determining that the memory required for the reduced scope of the data processing task is smaller than the amount of available memory; and

scheduling the reduced scope of the data processing task with the scheduler; and

in response to the scheduling, storing the data records using the available memory and performing the reduced scope of the data processing task to process the data records.

8. The computer system of claim 7, wherein the factory object comprises program instructions that when executed, cause the computer processor to produce the data processing task.

9. The computer system of claim 7, wherein the factory object has a memory footprint smaller than a memory footprint of the data processing task.

10. The computer system of claim 7, wherein the operations further comprise:

in response to determining, at the first time, that the memory is not available for storing the data records for performing the data processing task, placing the data processing task into a queue of data processing tasks; and

in response to determining, at the second time, that the memory is available for performing the reduced scope of the data processing task, removing the data processing task from the queue of data processing tasks.

11. The computer system of claim 10, wherein the operations further comprise:

determining, at a third time that is after the first time but before the second time, whether the memory is available for performing the data processing task; and

in response to determining, at the third time, that the memory is not available for performing the data processing task, placing the data processing task back into the queue.

12. The computer system of claim 7, wherein determining, at the first time, that the memory is not available for storing the data records for performing the data processing task comprises:

determining an amount of memory required for performing the data processing task;

determining an amount of memory that has not been reserved for other data processing tasks;

determining whether the amount of memory required for performing the data processing task is more than the amount of memory that has not been reserved for other data processing tasks; and

in response to determining that the amount of memory required for performing the data processing task is more than the amount of memory that has not been reserved for other data processing tasks, determining that the memory is not available for performing the data processing task.

13. A non-transitory computer readable storage medium storing executable computer program instructions for managing memory in a computer, the executable computer program instructions comprising instructions that when executed cause a computer processor to perform operations comprising:

accessing an analytics workflow comprising a sequence of tools, each tool in the sequence of tools adapted to perform a data analytics function;

determining, at a tool in the analytics workflow at a first time, whether a memory associated with the computer is available for storing data records for performing a data processing task;

in response to determining, at the first time, that the memory is not available for storing data records for performing the data processing task, generating a factory object for the data processing task, the factory object specifying instructions to produce the data processing task;

registering, by the tool, the factory object with a scheduler;

in response to detection of a free thread by the scheduler, requesting a task from the factory object;

in response to the request, producing the data processing task by using the factory object;

determining, at a second time that is after the first time, that memory is available for performing a reduced scope of the data processing task and scheduling the reduced scope of the data processing task with the scheduler, the determining comprising:

determining that the memory for storing the data records of the data processing task is more than an amount of available memory;

determining a smaller amount of memory required for performing the reduced scope of the data processing task;

determining that the memory required for the reduced scope of the data processing task is smaller than the amount of available memory; and

scheduling the reduced scope of the data processing task with the scheduler; and

in response to the scheduling, storing the data records using the available memory and performing the reduced scope of the data processing task to process the data records.

14. The non-transitory computer readable storage medium of claim 13, wherein the factory object comprises program instructions that when executed, cause the computer processor to produce the data processing task.

15. The non-transitory computer readable storage medium of claim 13, wherein the factory object has a memory footprint smaller than a memory footprint of the data processing task.

16. The non-transitory computer readable storage medium of claim 13, wherein the operations further comprise:

in response to determining, at the first time, that the memory is not available for storing the data records for performing the data processing task, placing the data processing task into a queue of data processing tasks; and

in response to determining, at the second time, that the memory is available for performing the reduced scope of the data processing task, removing the data processing task from the queue of data processing tasks.

17. The non-transitory computer readable storage medium of claim 13, wherein determining, at the first time, that the memory is not available for storing the data records for performing the data processing task comprises:

determining an amount of memory required for performing the data processing task;

determining an amount of memory that has not been reserved for other data processing tasks;

determining whether the amount of memory required for performing the data processing task is more than the amount of memory that has not been reserved for other data processing tasks; and

in response to determining that the amount of memory required for performing the data processing task is more than the amount of memory that has not been reserved for other data processing tasks, determining that the memory is not available for performing the data processing task.