as a same memory and processor for implementing instructions from the memory. Programs may be parts of a single program, separate programs, or distributed across several memories and processors.

In some examples, when it is said that an interface is performing some action, the action may be performed by the interface directly, and/or the action may be performed by a logic which provides and/or implements the interface, such as by an interface-implementing logic. The interface-implementing logic may be any logic which provides the interface and/or implements the logic that performs actions described as being performed by the interface. The interface-implementing logic may be and/or may include any of the logics described herein. For example, if the actions described as being performed by one or more data interfaces are performed by the client logic **312**, then the interface-implementing logic for the corresponding data interface(s) may be and/or may include the client logic. In examples where the action is performed by the interface, the actions performed may be performed in response to an invocation of the interface. In examples where the action is performed by the interface-implementing logic, the actions performed may be performed in response to an invocation of the corresponding interface.

The respective logic, software or instructions for implementing the processes, methods and/or techniques discussed throughout this disclosure may be provided on computer-readable media or memories or other tangible media, such as a cache, buffer, RAM, removable media, hard drive, other computer readable storage media, or any other tangible media or any combination thereof. The tangible media include various types of volatile and nonvolatile storage media. The functions, acts or tasks illustrated in the figures or described herein may be executed in response to one or more sets of logic or instructions stored in or on computer readable media. The functions, acts or tasks are independent of the particular type of instructions set, storage media, processor or processing strategy and may be performed by software, hardware, integrated circuits, firmware, micro code, or any type of other processor, operating alone or in combination. Likewise, processing strategies may include multiprocessing, multitasking, parallel processing and/or any other processing strategy known now or later discovered. In one embodiment, the instructions are stored on a removable media device for reading by local or remote systems. In other embodiments, the logic or instructions are stored in a remote location for transfer through a computer network or over telephone lines. In yet other embodiments, the logic or instructions are stored within a given computer, CPU, GPU, or system.

The logic illustrated in the flow diagrams may include additional, different, or fewer operations than illustrated. The operations illustrated may be performed in an order different than illustrated.

A second action may be said to be "in response to" a first action independent of whether the second action results directly or indirectly from the first action. The second action may occur at a substantially later time than the first action and still be in response to the first action. Similarly, the second action may be said to be in response to the first action even if intervening actions take place between the first action and the second action, and even if one or more of the intervening actions directly cause the second action to be performed. For example, a second action may be in response to a first action if the first action sets a flag and a third action later initiates the second action whenever the flag is set.

To clarify the use of and to hereby provide notice to the public, the phrases "at least one of <A>, <B>, . . . and <N>" or "at least one of <A>, <B>, <N>, or combinations thereof" or "<A>, <B>, . . . and/or <N>" are defined by the Applicant in the broadest sense, superseding any other implied definitions hereinbefore or hereinafter unless expressly asserted by the Applicant to the contrary, to mean one or more elements selected from the group comprising A, B, . . . and N. In other words, the phrases mean any combination of one or more of the elements A, B, . . . or N including any one element alone or the one element in combination with one or more of the other elements which may also include, in combination, additional elements not listed. Unless otherwise indicated or the context suggests otherwise, as used herein, "a" or "an" means "at least one" or "one or more."

While various embodiments have been described, it will be apparent to those of ordinary skill in the art that many more embodiments and implementations are possible. Accordingly, the embodiments described herein are examples, not the only possible embodiments and implementations.

What is claimed is:

1. A method comprising:

mapping a portion of a local primary memory to an address space of a virtualization instance at a client device, wherein the address space is addressable by a processor of the client device, and wherein the virtualization instance is executed via the processor of the client device;

mapping a first portion of external memory to the address space, wherein the external memory is memory that is external to the client device but treated as primary memory at the client device, and wherein a second portion of the external memory is not in any address space addressable by the processor of the client device; and

providing a performance indication data structure indicating a performance difference between the portion of the local primary memory mapped to the address space and the first portion of external memory mapped to the address space; and

causing data to migrate between the external memory and the local primary memory based on the performance indication data structure by the virtualization instance, wherein the data is caused to migrate by the virtualization instance and/or a logic configured to operate with the virtualization instance,

wherein the performance indication data structure indicates that the local primary memory is associated with a first virtualization instance architectural element and the external primary memory is associated with a second virtualization instance architectural element.

2. The method of claim **1**, further comprising caching data from the first portion of the external memory in a second portion of the local primary memory.

3. The method of claim **1**, wherein, in response to the migration of data, a second portion of the local primary memory holds data previously held in the external memory or wherein the second portion of the external memory and/or a third portion of the external memory mapped to the address space holds data previously held in the portion of the local primary memory.

4. The method of claim **1**, wherein the virtualization instance and/or the logic configured to operate with the virtualization instance causes the data to be migrated in response to access of the data.

5. The method of claim 1, wherein the data is migrated without copying the data by reassociating the portion of local primary memory holding the data from one mapping to another.

6. The method of claim 1, wherein the external memory is included in a memory appliance and wherein data of the external memory is accessible via client-side memory access in which a communication interface of the memory appliance is configured to access the external memory in the memory appliance.

7. The method of claim 1, wherein one or more mappings to the address space are created, destroyed, modified, and/or updated during operation of the virtualization instance.

8. The method of claim 1, wherein the performance indication data structure is created, destroyed, modified, and/or updated during operation of the virtualization instance.

9. The method of claim 1, wherein the performance indication data structure indicates one or more relative and/or absolute performance attributes of the local primary memory, the external primary memory, and/or one or more mappings to the address space.

10. The method of claim 1, wherein the first virtualization instance architectural element and the second virtualization instance architectural element are Non-Uniform Memory Architecture nodes, ACPI Proximity Domains, and/or ACPI Memory Proximity Domains.

11. The method of claim 1, wherein the performance indication data structure includes ACPI System Resource Affinity Table information, ACPI Static Resource Affinity Table information, ACPI Heterogeneous Memory Attributes, ACPI Heterogeneous Memory Attributes information, and/or ACPI Memory Proximity Domain Attributes Structure information.

12. The method of claim 1, wherein the logic configured to operate with the virtualization instance causes the data to migrate to memory with faster performance for frequently-accessed data and/or recently-accessed data, and/or wherein the logic configured to operate with the virtualization instance causes the data to migrate to memory with slower performance for infrequently-accessed data and/or not-recently-accessed data.

13. The method of claim 1, wherein the logic configured to operate with the virtualization instance causes the data to migrate to memory with a shorter distance for frequently-accessed data and/or recently-accessed data, and/or wherein the logic configured to operate with the virtualization instance causes the data to migrate to memory with a longer distance for infrequently-accessed data and/or not-recently-accessed data.

14. A method, comprising:

mapping a portion of a local primary memory to an address space of a virtualization instance at a client device, wherein the address space is addressable by a processor of the client device, and wherein the virtualization instance is executed via the processor of the client device;

mapping a first portion of external memory to the address space, wherein the external memory is memory that is external to the client device but treated as primary memory at the client device, and wherein a second portion of the external memory is not in any address space addressable by the processor of the client device; and

providing a performance indication data structure indicating a performance difference between the portion of the local primary memory mapped to the address space and the first portion of external memory mapped to the address space; and causing data to migrate between the external memory and the local primary memory based on the performance indication data structure by the virtualization instance, wherein the data is caused to migrate by the virtualization instance and/or a logic configured to operate with the virtualization instance,

wherein the mapping of the first portion of the external memory creates a first mapping of external memory, and wherein the performance indication data structure indicates a performance difference between the first mapping of external memory and a second mapping of external memory.

15. The method of claim 14, wherein the first mapping of external memory maps one or more portions of memory of a first memory appliance and the second mapping of external memory maps one or more portions of memory of a second memory appliance.

16. The method of claim 14, wherein the first mapping of external memory maps one or more portions of a first memory of a memory appliance and the second mapping of external memory maps one or more portions of a second memory of the memory appliance.

17. The method of claim 1, wherein in response to a low memory condition at the client device, an amount of the local primary memory used to hold one or more cached copies of data of the external primary memory is reduced and an amount of the local primary memory backing the mapping of local primary memory is not reduced or is reduced by a lesser extent than the amount of the local primary memory used to hold the one or more cached copies of data of the external primary memory.

18. The method of claim 1, wherein an amount of the local primary memory mapped to the address space and/or an amount of the local primary memory used to cache data in the external primary memory is determined based on memory access behavior of the logic configured to with the virtualization instance.

19. The method of claim 1, wherein the external primary memory is added to the virtualization instance in response to a memory hot-add event.

20. The method of claim 1, wherein the external primary memory is removed from the virtualization instance in response to a memory hot-remove event.

21. The method of claim 1, wherein the local primary memory and/or the external primary memory is added to and/or removed from the virtualization instance using memory ballooning.

22. The method of claim 1, wherein the causing data to migrate between the external memory and the local primary memory is based on the performance indication data structure indicating a source portion having a lower power draw and/or higher power draw than a destination portion.

* * * * *