CLAIMS

1. A scalable container-based system implemented in computer instructions stored in a non-transitory medium, the system comprising:

a source code repository containing the source code of a monolithic legacy application containing a plurality of programs executable in a legacy computing environment to perform a plurality of transactions;

a source code analyzer operable to parse the source code and to identify, for each transaction in the plurality of transactions a transaction definition vector identifying each program potentially called during the transaction, to create a plurality of transaction definition vectors;

a transaction state definition repository operable to store the plurality of transaction definition vectors;

an activity log analyzer operable to create a dynamic definition repository identifying which programs are actually used by the monolithic legacy application in performing in at least a subset of the plurality of transactions;

a microservice definition optimizer operable to compare the plurality of transaction definition vectors to the dynamic definition repository and remove unused programs from the plurality of transaction definition vectors to arrive at a corresponding plurality of microservice definition vectors defining a plurality of microservices;

a microservice image builder operable to, for each microservice definition vector of the plurality of microservice definition vectors, locate for each program identified by the microservice definition vector compiled source code binaries compiled to run in the legacy computing environment and to combine, for each microservice definition vector of the plurality of microservice definition vectors, corresponding compiled binaries to generate a microservice image to form a plurality of microservce images;

a microservice image repository operable to store the plurality of microservice images;

a complementary component repository operable to store a set of binary images of emulator elements of a legacy emulator that, together, are less than a

complete legacy emulator, said images corresponding to a plurality of functions or sets of functions of said legacy computing environment, and said images executable in a distinct computer environment characterized by an instruction set distinct from the instruction set of the legacy environment;

a container builder operable to form a container image for each microservice or a set of microservices in the plurality of microservices using the corresponding microservice image or images from the microservice image repository and using image files from the complementary component repository for the emulator elements of the legacy emulator corresponding to functions or sets of functions employed by the microservice or set of microservices when executed, as identified by signatures of calls in the binaries in the microservice or set of microservices, to create a plurality of container images;

a container image repository operable to store the plurality of container images executable in the distinct computing environment; and

a container management system operable to create at least one container for execution in the distinct computing environment and to run at least one container image stored in the container image repository in the at least one container.

2. The scalable container-based system of Claim 1, wherein the activity log analyzer is operable to create a plurality of dynamic transaction definition vectors that correspond to at least a portion of the plurality of transaction definition vectors, and wherein the microservice definition optimizer compares each dynamic transaction definition vector to each corresponding transaction definition vector to create the plurality of microservice definition vectors; and/or

wherein the activity log analyzer uses legacy activity logs of the monolithic legacy application generated by running the monolithic legacy application in the legacy computing environment.

3. The scalable container-based system of any one of the preceding claims wherein the activity log analyzer uses an emulator to run the monolithic

legacy application to generate log files and to determine which programs are used by the monolithic legacy application during the execution of transactions; and/or

wherein the source code analyzer is operable to use information from the activity log analyzer to identify the transaction definition vectors; and/or

wherein the source code analyzer is further operable to create a plurality of translation tables.

4. The scalable container-based system of any one of the preceding claims, wherein the microservice definition optimizer is operable to further optimize the microservice definition vectors, and

wherein optionally the microservice definition optimizer is operable to further optimize the microservice definition vectors by creating additional microservice definition vectors containing programs shared by more than one transaction in the plurality of transactions.

5. The scalable container-based system of any one of the preceding claims, further comprising a binary repository operable to store the compiled source code containing binaries compiled to run in the legacy computing environment, and

wherein optionally the compiled source code in the binary repository is compiled from the source code in the source code repository into binary files.

6. The scalable container-based system of any one of the preceding claims, wherein the legacy computing environment comprises a Multiple Virtual Storage (MVS) or z/OS computer system; and/or

wherein the complementary component repository is further operable to store a plurality of images of operating system software packages used by the legacy emulator, and wherein the container builder also places images of any software packages used by a particular element of the legacy emulator in a particular container image containing the particular element of the legacy emulator; and/or

wherein the container builder is further operable to replace the signatures of calls in the binaries in the microservice or set of microservices with instructions for calls operable in the legacy emulator.

7. The scalable container-based system of any one of the preceding claims, wherein the container management system is operable to create a plurality of containers, and

wherein optionally a set of complementary images are instantiated in a separate container within a common pod.

8. The scalable container-based system of any of claim 7, wherein more than one copies of at least one container image are activated in more than one separate containers; and/or

wherein the container management system is operable to vary the number of containers in the plurality of containers; and/or

wherein the container management system is operable to allocate varying resources to separate containers; and/or

wherein the container management system is operable to use information from the activity log analyzer to determine how the number of copies of at least one container image is to place into more than one separate containers, to determine the number of containers in the plurality of containers, and/or to determine resources to allocate to separate containers; and/or

wherein the container management system is operable to use information from use of the scalable container-based system to determine how the number of copies of at least one container image is to place into more than one separate containers, to determine the number of containers in the plurality of containers, and/or to determine resources to allocate to separate containers.

9. The scalable container-based system of any one of the preceding claims, wherein the source code analyzer is further operable to create one or more

sub-databases or clusters of sub-databases from a database of the monolithic legacy application; and/or

wherein the container builder is operable to place the one or more sub-databases or clusters of sub-databases in one or more containers; and/or

wherein, when the source code is changed, the container-based system is operable to automatically update at least one microservice image, at least one container image, and at least one container to contain an updated binary based on the source code change.

10. A method of creating and operating a scalable container-based system, the method comprising:

parsing a monolithic legacy application executable in a legacy computing environment and partitioning its program files to create a plurality of transaction definition vectors corresponding to a plurality of transactions performable by the monolithic legacy application and identifying, for each transaction, all programs called by that transaction;

storing the plurality of transaction definition vectors in a transaction state repository;

for at least a portion of the plurality of transactions, creating a dynamic definition repository by determining which programs are actually used when the transaction is performed by the monolithic legacy application;

comparing the plurality of transaction definition vectors to the dynamic definition repository and removing programs not used in a transaction from its corresponding transaction definition vector to arrive at a corresponding microservice definition vector to create a plurality of microservice definition vectors;

for each microservice definition vector of the plurality of microservice definition vectors, locating corresponding compiled source code binaries compiled to run in the legacy computing environment and combining said binaries for generating a microservice image containing the corresponding compiled source code to form a plurality of microservice images;

storing the plurality of microservice images in a microservice image repository;

storing, in a complementary component repository, images of a plurality of elements a legacy emulator operable to execute programs in a different computing environment than the legacy computing environment, the elements of the legacy emulator corresponding to a plurality of functions or sets of functions of the monolithic legacy application;

forming a container image for each microservice or a set of microservices in the plurality of microservices using the corresponding microservice image or images from the microservice image repository and using image files from the complementary component repository for the elements of the legacy emulator corresponding to functions or sets of functions employed by the microservice or set of microservices when executed, as identified by signatures of calls in the binaries in the microservice or set of microservices, to create a plurality of container images;

storing the container images in a container image repository;

creating at least one container in the different computing environment using a container management system and storing at least one container image in the container in a form executable in the different computing environment; and

executing the at least one container in the container.

11. The method of Claim 10, comprising:

creating a plurality of dynamic transaction definition vectors that correspond to at least a portion of the plurality of transaction definition vectors using the activity log analyzer, and

comparing each dynamic transaction definition vector to each corresponding transaction definition vector to create the plurality of microservice definition vectors using the microservice definition optimizer; and/or

comprising the activity log analyzer using legacy activity logs of the monolithic legacy application generated by running the monolithic legacy application in the legacy computing environment.

12. The method of claim 10 or 11, comprising the activity log analyzer using an emulator to run the monolithic legacy application to generate log files and to determine which programs are used by the monolithic legacy application during the execution of transactions; and/or

comprising the source code analyzer using information from the activity log analyzer to identify the transaction definition vectors; and/or

creating a plurality of translation tables using the source code analyzer.

13. The method of any one of claims 10-12, comprising further optimizing the microservice definition vectors using the microservice definition optimizer, and

optionally comprising further optimizing the microservice definition vectors using the microservice definition optimizer by creating additional microservice definition vectors containing programs shared by more than one transaction in the plurality of transactions.

14. The method of any one of claims 10-13, further comprising storing the compiled source code containing binaries compiled to run in the legacy computing environment in a binary repository, and

optionally comprising compiling the source code in the binary repository from the source code in the source code repository into binary files.

15. The method of any one of claims 10-14, wherein the legacy computing environment comprises a Multiple Virtual Storage (MVS) or z/OS computer system; and/or

comprising the complementary component repository storing a plurality of images of operating system software packages used by the legacy emulator, and the container builder also placing images of any software packages used by a particular element of the legacy emulator in a particular container image containing the particular element of the legacy emulator; and/or

comprising the container builder replacing the signatures of calls in the binaries in the microservice or set of microservices with instructions for calls operable in the legacy emulator.

16. The method of any one of claims 10-15, comprising creating a plurality of containers using the container management system, and

optionally comprising instantiating a set of complementary images in a separate container within a common pod.

17. The method of claim 16, comprising activating more than one copies of at least one container image in more than one separate containers; and/or

comprising the container management system varying the number of containers in the plurality of containers; and/or

comprising the container management system allocating varying resources to separate containers; and/or

comprising the container management system using information from the activity log analyzer to determine how the number of copies of at least one container image is to place into more than one separate containers, to determine the number of containers in the plurality of containers, and/or to determine resources to allocate to separate containers; and/or

comprising the container management system using information from use of the scalable container-based system to determine how the number of copies of at least one container image to place into more than one separate containers, to determine the number of containers in the plurality of containers, and/or to determine resources to allocate to separate containers.

18. The method of any one of claims 10-17, comprising the source code analyzer creating one or more sub-databases or clusters of sub-databases from a database of the monolithic legacy application; and/or

comprising the container builder placing the one or more sub-databases or clusters of sub-databases in one or more containers; and/or

comprising, when the source code is changed, automatically updating at least one microservice image, at least one container image, and at least one container to contain an updated binary based on the source code change.