A. Abstract
I have written a simple database which I then compile one version for Linux and another for Nautilus.

B. Description
1) Check-list (artifact meta information):
   • Program: db-multiverse
   • Run-time environment: Linux 4.17.6 (in Fedora Release 28) and Nautilus @ commit 2fb4e52816
   • Hardware: AMD EPYC 7281 16-Core Processor
   • Compiler: gcc 8.1.1 20180712 (Red Hat 8.1.1-5).
   • Compiler Options: See the Makefile and the nautilus/Makefile for compile-options.
   • Experiment workflow: See section.
   • Experiment customization: Modify what work the database is doing, the number of chunks, number of columns, chunk-size, and domain-size (domain of the elements in the database).
   • Publicly available: Yes
2) How software can be obtained (if available): NautDB can be cloned from this GitHub Repository.
3) Hardware dependencies: At the time of this writing, Nautilus supports x86_64, Xeon Phi, and the GEM5 simulator.
   In order to gather performance data, I use the CPU-enabled performance counters. These are specific within processor families. I have targetted 'AMD EPYC 7281 16-Core Processor', but the code can be modified for other processors as well (see src/app/perf* and include/app/perf*). If this is not modified, you can still collect cycle-counts.
4) Software dependencies: gcc, GNU make, GNU libc (for running in Linux), grub2 (for compiling Nautilus), xorriso (for compiling Nautilus)

C. Installation
1) Download the source code.
   git clone git@github.com:
   HExSA-Lab/db-multiverse.git

D. Experiment workflow
1) Manual Workflow:
   1) Compile the Linux version
      make main
   2) Run and collect output
      ./main > linux_output
   3) Compile Nautilus
      ./scripts/insert_into_nautilus.sh
      make -C nautilus nautilus.bin
   4) Boot into Nautilus. If you are using grub,
      mv nautilus.bin /boot/nautilus.bin
      echo <<EOF
        menuentry "Nautilus" {
          # adjust for your specific hw
          set root='hd0,msdos1'
          multiboot2 /nautilus.bin
          boot
        }
        EOF
        >> /etc/grub.d/41_custom
        reboot
        # wait for grub menu
        # select Nautilus from the menu
        # capture output over serial link
   2) Workflow Automation: This is the simplest way of reproducing the Nautilus experiment. However, the user can benefit from automation.
      • A hardware management tool such as IPMI can automate the process of rebooting, selecting Nautilus from the Grub menu, and capturing the serial remotely. This is implemented in scripts/ipmi_helper.sh.
      • expect can be used to automate navigating the Grub menu. This is implemented in scripts/drive_grub.py.
      • See scripts/run_linux.sh and scripts/run_nautk.sh for start-to-finish automation on both platforms with a remote run-host and a remote build-host.

E. Evaluation and expected result
The software will output blocks of CSV data wrapped in curly-braces, such as:

```markdown
file: cool_data.csv {
  x column1 name,column2 name,title
  1,2,
}
```

The independent variables have a header beginning with ‘x’.

F. Experiment customization
Customize src/app/main.c to choose which modules to run.
To customize the test_db module, edit src/app/test_db.c. The parameters for the database (number of columns, chunk size, number of chunks, domain size) and which operators will be timed can be customized here.

G. Notes
It may be tempting to run Nautilus in a virtualized environment such as QEMU instead of on bare-metal. While this can be an error-detecting step, but the virtualized environment will not have yield realistic performance data.