Challenges of using Containers to Run Graphical Embedded Systems

Diego Rondini

$ whoami

Diego Rondini

- embedded engineer at Kynetics
- worked on lots of Android and Linux OSes

What this session is about

Providing an overview

of the possible approaches

to enjoy containers benefits

on embedded Linux plaftorm

running graphical applications

Poll

Who of you is already using containers / docker?

Who of you is already using containers
on embedded boards?

Project target

- explore what is available in the open source space to run graphical applications on Linux

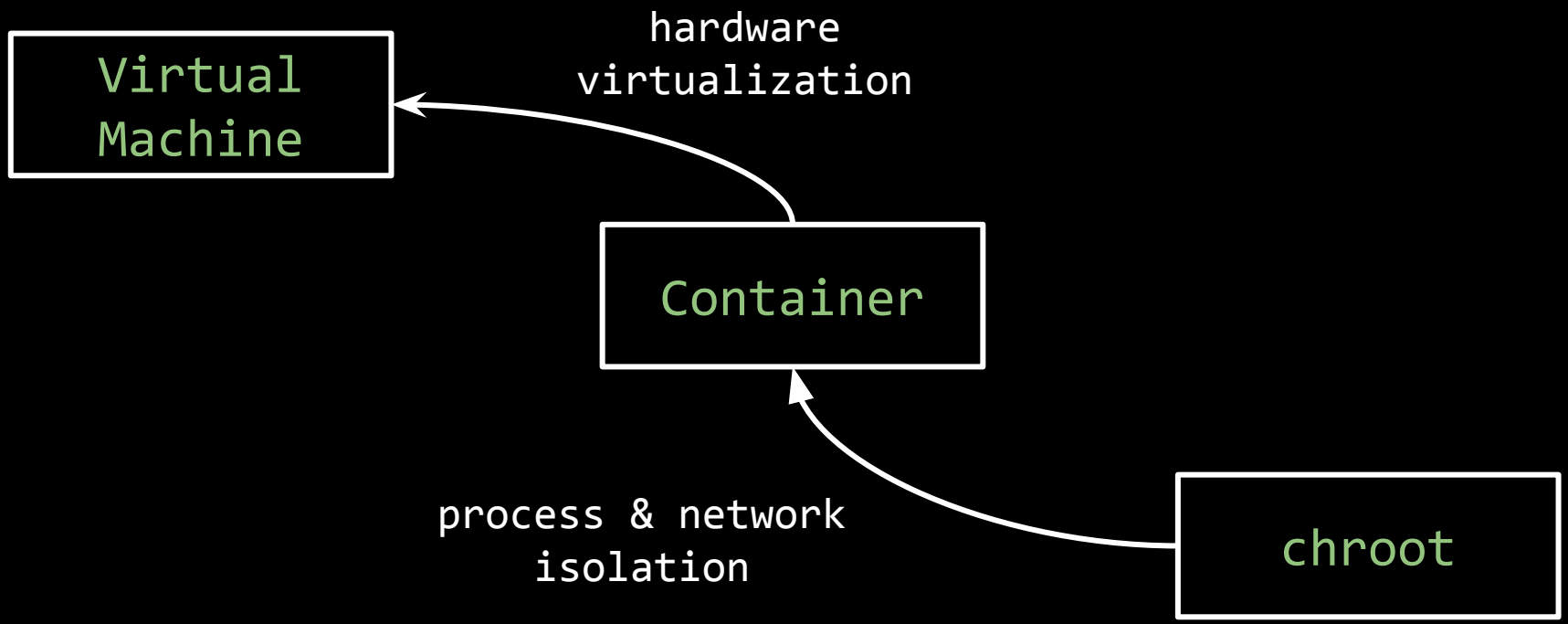
Agenda

- Motivations
- Which options do we have?
- Impact of hardware and drivers (esp. 3D)
- Performance impact
- Security considerations

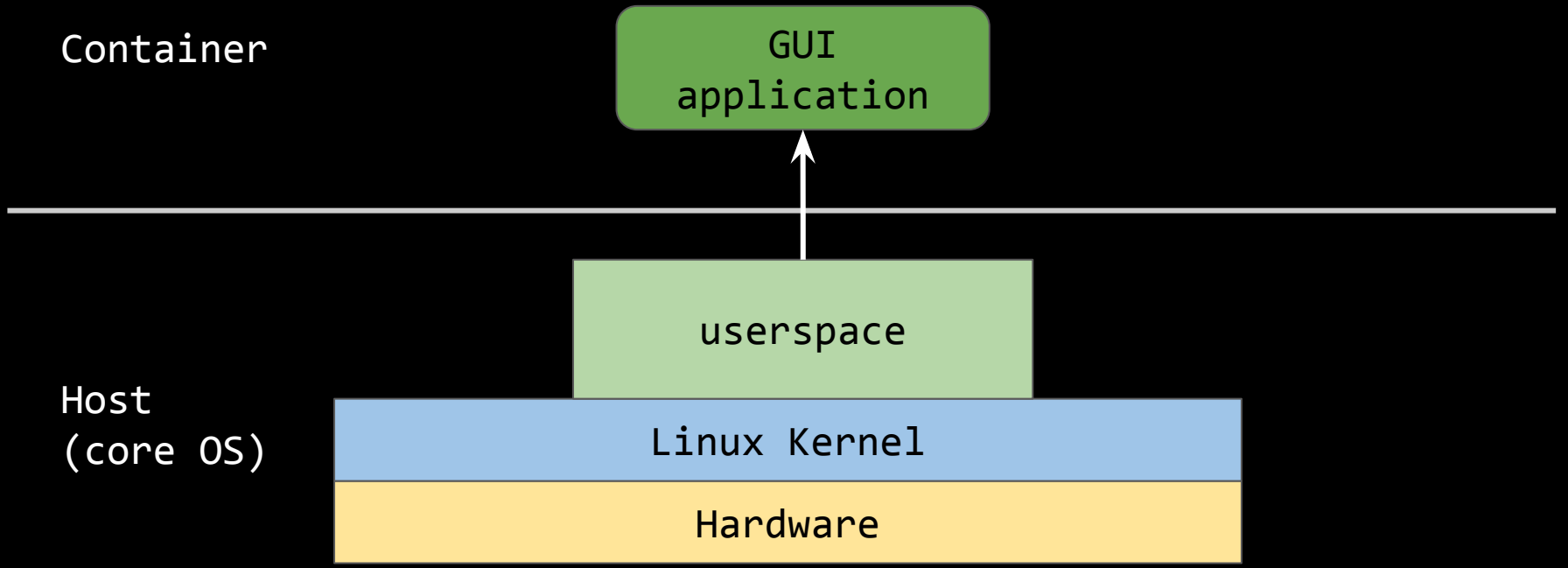
Motivations

- Avoiding some of the hassles of building everything from sources (Yocto anyone?)
- Packaging target applications quickly and consistently
- Including all userspace dependencies
- Improving portability

What are containers?



Host vs container



How to run GUI app in a container

- use network remote protocol
- use local IPC (socket sharing)
- run display server on the host and client window on the container
- run display server on the container

How to run X11 app in a container

A. X11 display server on the host + X11 application over network remote protocol (VNC, xpra, …) in the container
B. X11 display server on the host + X11 application as X11 client sharing host X11 socket in the container
C. X11 display server on the host + X11 application as client of X-on-X implementation (Xephyr) in the container
D. Wayland compositor on the host + X11 application as XWayland client in the container
E. X11 display server and X11 application in the container

How to run Wayland app in a container

A. Wayland compositor on the host + Wayland application client in the container
B. Wayland compositor and Wayland application client in the container
C. X11 display server on the host + Wayland compositor and Wayland application client in the container (do you really want to?!?)

Remoting option - xpra (X11.A)

Xpra is an open-source multi-platform persistent remote display server and client for forwarding applications and desktop screens.

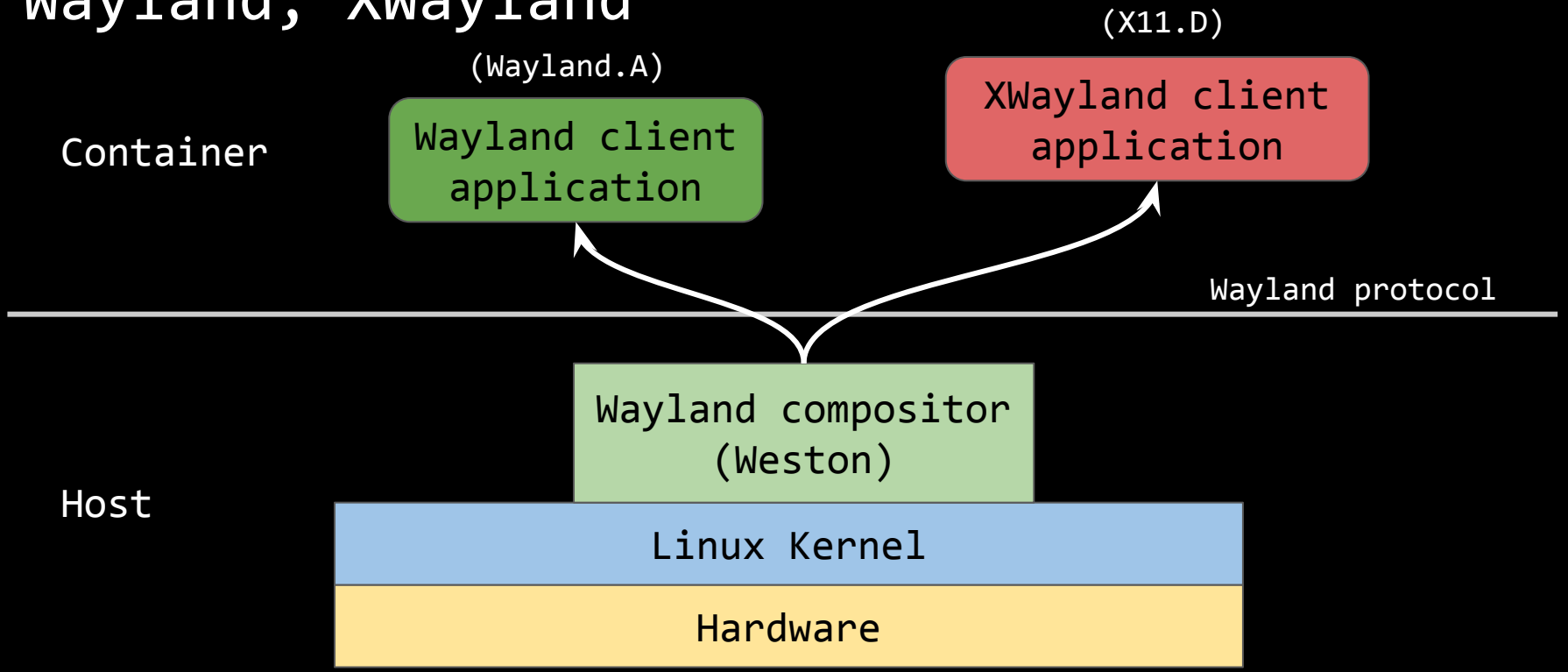
+ Good isolation
+ supports detaching (“screen for X11”)
- no support for 3D GPU accel

Remoting option - Xephyr (X11.C)

Xephyr is an X-on-X implementation that targets a window on a host X Server.

+ Good isolation, effectively running on a dedicate X Server
- no support for 3D GPU accel

Wayland, XWayland



x11docker

x11docker is a shell script that allows to run graphical desktop applications (and entire desktops) in Docker Linux containers

https://github.com/mviereck/x11docker

+ simple to setup (just 1 script)
+ simple to try things out
+ very well documented
+ lots of different options

Hardware

Open source graphics stack

- Toradex Apalis i.MX6
- Vivante GC2000
- mSATA disk

Proprietary graphics stack

- Boundary Nitrogen8M i.MX8M
- Vivante GC7000 Lite

Open source graphics stack

We used Fedora 30 for ARM with etnaviv driver to test a fully open source graphics stack.

https://fedoraproject.org/wiki/Architectures/ARM

Fedora 30 on Apalis i.MX6 - basic boot

1. install an mSATA disk
2. install Fedora for ARM on the disk
3. adjust U-Boot to boot from SATA

```
=> setenv soc imx6q
=> setenv board apalis-ixora-v1.1
=> setenv bootcmd "run sata_boot; ${bootcmd}"
=> saveenv
=> reset
```

Fedora 30 on Apalis i.MX6 - enable video

1. tweak kernel parameters
   a. to use HDMI disable other outputs:
      video=DPI-1:d video=LVDS-1:d
   b. with recent kernels (5.0, 5.1, 5.2) put CMA area where etnaviv expects it (see https://lists.freedesktop.org/archives/dri-devel/2019-June/223516.html):
      cma=256M@2G (at the end of RAM size)
2. install “armada” X11 2D driver to use X11:
   # dnf install xorg-x11-drv-armada

Fedora 30 on Apalis i.MX6 - Weston

3. Weston 6.0 has a bug that closes the session when it should not (“deactivating session” on the Weston startup log). Fix is building Weston 7.0-alpha or backporting:
   a. https://gitlab.freedesktop.org/wayland/weston/commit/c569bdc23612eab518b6f60cf60fb3ab775cf5e4
   b. https://gitlab.freedesktop.org/wayland/weston/commit/49dc32013eb7d65a3aa154dc3f821ae31a8696bd

Mesa “mesa-dri-drivers” 19.x already includes required 3D drivers to have OpenGL 2 / OpenGL ES2 acceleration.

Fedora 30 on Apalis i.MX6 - build Weston

```
# dnf builddep weston
# dnf install git

$ git clone https://gitlab.freedesktop.org/wayland/weston.git
$ meson build/ --prefix=/usr/local -Dremoting=false
-Dpipewire=false -Dsimple-dmabuf-drm=etnaviv
$ ninja -C build/
# ninja -C build/ install

$ LD_LIBRARY_PATH=/usr/local/lib weston -i 0
```

Using etnaviv on i.MX6 in Docker

Dockerfile

```
FROM ubuntu:19.04
ARG DEBIAN_FRONTEND=noninteractive
RUN apt-get update && apt-get install -y x11-apps weston glmark2 glmark2-es2 glmark2-wayland
glmark2-es2-wayland
```

```
docker image build --tag ubuntu-x11-wayland:1 .
```

Using etnaviv on i.MX6 in Docker

X11 running on the host with glmark2 (OpenGL 2) X11 client running in Docker (X11.B):

```
$ x11docker --hostdisplay --gpu --share /home/fedora/glmark/glmark.conf:ro -- ubuntu-x11-wayland:1
glmark2 \-f /home/fedora/glmark/glmark.conf
```

Weston running on the host with glmark2-es2-wayland (OpenGL ES2) Wayland client running in Docker (Wayland.A):

```
$ x11docker --hostwayland --gpu --share /home/fedora/glmark/glmark.conf:ro -- ubuntu-x11-wayland:1
glmark2-es2-wayland \-f /home/fedora/glmark/glmark.conf
```

Weston and glmark2-es2-wayland (OpenGL ES2) Wayland client running in Docker (Wayland.B):

```
$ x11docker --weston --gpu --share /home/fedora/glmark/glmark.conf:ro -- ubuntu-x11-wayland:1
glmark2-es2-wayland \-f /home/fedora/glmark/glmark.conf
```

Proprietary graphics stack

To test the Vivante proprietary graphics stack we decided to run Ubuntu 18.04 with Weston for Nitrogen8M.

Installation instructions at:
https://boundarydevices.com/ubuntu-bionic-18-04-lts-for-nitrogen8m-board-june-2019-kernel-4-14-x/

Using Vivante on i.MX8QM in docker

Dockerfile

```
FROM ubuntu:18.04
COPY apt/ /etc/apt/
ARG DEBIAN_FRONTEND=noninteractive
RUN apt-get update && apt-get dist-upgrade -y && apt-get install -y dpkg-dev imx-gpu-viv-b16-all
RUN apt-get install -y glmark2-es2-wayland mesa-utils-extra
RUN update-alternatives --install /usr/lib/aarch64-linux-gnu/libEGL.so aarch64-linux-gnu_libEGL
/usr/lib/aarch64-linux-gnu/vivante/libEGL.so.1.0 1000
RUN update-alternatives --install /usr/lib/aarch64-linux-gnu/libGLESv2.so
aarch64-linux-gnu_libGLESv2 /usr/lib/aarch64-linux-gnu/vivante/libGLESv2.so.2 1000
RUN ldconfig
```

Using Vivante on i.MX8QM in docker

Weston running on the host with glmark2-es2-wayland (OpenGL ES2) Wayland client running in Docker (Wayland.A):

```
$ x11docker --hostwayland --debug --gpu --share /dev/galcore --share
/home/ubuntu/glmark/glmark.conf:ro -- ubuntu18.04-glmark:1 glmark2-es2-wayland \-f
/home/ubuntu/glmark/glmark.conf
```

Reference glmark.conf:

```
build:use-vbo=true
texture:texture-filter=mipmap
shading:shading=cel
jellyfish
```

Performance – X11 with etnaviv (i.MX6Q)

❯ Host (Fedora 30)

```
=======================================================
    glmark2 2017.07
=======================================================
    OpenGL Information
    GL_VENDOR:    etnaviv
    GL_RENDERER:   Vivante GC2000 rev 5108
    GL_VERSION:  2.1 Mesa 19.1.3
=======================================================
[build] use-vbo=true: FPS: 106 FrameTime: 9.434 ms
[texture] texture-filter=mipmap: FPS: 79 FrameTime:
12.658 ms
[shading] shading=cel: FPS: 83 FrameTime: 12.048 ms
[jellyfish] <default>: FPS: 58 FrameTime: 17.241 ms
=======================================================
                                 glmark2 Score: 81
=======================================================
```

❯ Docker (Ubuntu 19.04)

```
=======================================================
    glmark2 2014.03+git20150611.fa71af2d
=======================================================
    OpenGL Information
    GL_VENDOR:    etnaviv
    GL_RENDERER:   Vivante GC2000 rev 5108
    GL_VERSION:  2.1 Mesa 19.0.2
=======================================================
[build] use-vbo=true: FPS: 104 FrameTime: 9.615 ms
[texture] texture-filter=mipmap: FPS: 79 FrameTime:
12.658 ms
[shading] shading=cel: FPS: 83 FrameTime: 12.048 ms
[jellyfish] <default>: FPS: 58 FrameTime: 17.241 ms
=======================================================
                                 glmark2 Score: 81
=======================================================
```

Performance – Wayland with etnaviv (i.MX6Q)

❯ Host (Fedora 30)

```
=======================================================
    glmark2 2017.07
=======================================================
    OpenGL Information
    GL_VENDOR:    etnaviv
    GL_RENDERER:   Vivante GC2000 rev 5108
    GL_VERSION:  OpenGL ES 2.0 Mesa 19.1.3
=======================================================
[build] use-vbo=true: FPS: 492 FrameTime: 2.033 ms
[texture] texture-filter=mipmap: FPS: 161 FrameTime:
6.211 ms
[shading] shading=cel: FPS: 182 FrameTime: 5.495 ms
[jellyfish] <default>: FPS: 87 FrameTime: 11.494 ms
=======================================================
                                 glmark2 Score: 230
=======================================================
```

❯ Docker (Ubuntu 19.04)

```
=======================================================
    glmark2 2014.03+git20150611.fa71af2d
=======================================================
    OpenGL Information
    GL_VENDOR:     etnaviv
    GL_RENDERER:   Vivante GC2000 rev 5108
    GL_VERSION:    OpenGL ES 2.0 Mesa 19.0.2
=======================================================
[build] use-vbo=true: FPS: 474 FrameTime: 2.110 ms
[texture] texture-filter=mipmap: FPS: 158 FrameTime:
6.329 ms
[shading] shading=cel: FPS: 179 FrameTime: 5.587 ms
[jellyfish] <default>: FPS: 89 FrameTime: 11.236 ms
=======================================================
                                 glmark2 Score: 225
=======================================================
```

Performance – Wayland with Vivante (i.MX8M)

❯ Host (Ubuntu 18.04)

```
=======================================================
    glmark2 2017.07
=======================================================
    OpenGL Information
    GL_VENDOR:    Vivante Corporation
    GL_RENDERER:   Vivante GC7000L
    GL_VERSION:  OpenGL ES 3.1 V6.2.4.p4.190076
=======================================================
[build] use-vbo=true: FPS: 1284 FrameTime: 0.779 ms
[texture] texture-filter=mipmap: FPS: 1156 FrameTime:
0.865 ms
[shading] shading=cel: FPS: 709 FrameTime: 1.410 ms
[jellyfish] <default>: FPS: 252 FrameTime: 3.968 ms
=======================================================
                                 glmark2 Score: 850
=======================================================
```

❯ Docker (Ubuntu 18.04)

```
=======================================================
    glmark2 2017.07
=======================================================
    OpenGL Information
    GL_VENDOR:    Vivante Corporation
    GL_RENDERER:   Vivante GC7000L
    GL_VERSION:  OpenGL ES 3.1 V6.2.4.p4.190076
=======================================================
[build] use-vbo=true: FPS: 1418 FrameTime: 0.705 ms
[texture] texture-filter=mipmap: FPS: 1198 FrameTime:
0.835 ms
[shading] shading=cel: FPS: 726 FrameTime: 1.377 ms
[jellyfish] <default>: FPS: 257 FrameTime: 3.891 ms
=======================================================
                                 glmark2 Score: 899
=======================================================
```

Some simple security tips

- be aware of X security flaws
- don’t add host user to docker group
- don’t run the application inside the container as root
- share only the resources the application really needs
  - share as read-only if possible
- use the following options:
  - --cap-drop ALL
  - --security-opt no-new-privileges

Links

- https://medium.com/@SaravSun/running-gui-applications-inside-docker-containers-83d65c0db110
- x11docker: https://github.com/mviereck/x11docker/wiki
- SELinux and containers:
  https://danwalsh.livejournal.com/78312.html
  http://www.projectatomic.io/blog/2016/03/no-new-privs-docker/

Disclaimers: All logos are property of the respective owners.

Thanks for the support to:
Nicola La Gloria, Andrea Zoleo, Martin Viereck (x11docker), Peter Robinson (Fedora on ARM), Daniel Stone (Collabora), Lucas Stach (Pengutronix).

and...

Thanks to my daughter Marianna for the drawings!

Contacts:

USA Kynetics LLC 2040 Martin Ave, Santa Clara CA 95050 Ph: +1 (408) 475 7760	Italy Kynetics Srl Via Longhin, Padova (PD) 35129 Ph: +39 (049) 781 1091
info@kynetics.com | www.kynetics.com	